(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
10 May 2001 (10.05.2001)

PCT

(10) International Publication Number
WO 01/33796 A1

(51) International Patent Classification[7]: H04L 27/30

(21) International Application Number: PCT/US00/30472

(22) International Filing Date: 3 November 2000 (03.11.2000)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
60/163,833 5 November 1999 (05.11.1999) US

(71) Applicant (*for all designated States except US*): **ZEUS WIRELESS, INC.** [US/US]; 8325 Guilford Road, Columbia, MD 21046 (US).

(72) Inventors; and
(75) Inventors/Applicants (*for US only*): **SOUCY, Dean** [—/US]; 3284 Danmark Drive, Glenwood, MD 21738 (US). **REILLY, Edward, P.** [—/US]; 125 Warren Avenue, Baltimore, MD 21230 (US). **OPSENICA, Stanko** [—/US]; 2402 Mt. Vernon Avenue #1, Alexandria, VA 22301 (US). **COUNTS, Reginald** [—/US]; 4651 Willow Grove Drive, Ellicott City, MD 21042 (US). **OLSEN, Thomas, Henry** [—/US]; 1091 Cherry Orchard Road, Dover, PA 17315 (US).

(74) Agent: **MCKINNEY, J., Andrew, Jr.**; Adelberg, Rudow, Dorf, Hendler & Sameth, LLC, 600 Mercantile Bank & Trust Building, 2 Hopkins Plaza, Baltimore, MD 21201-2927 (US).

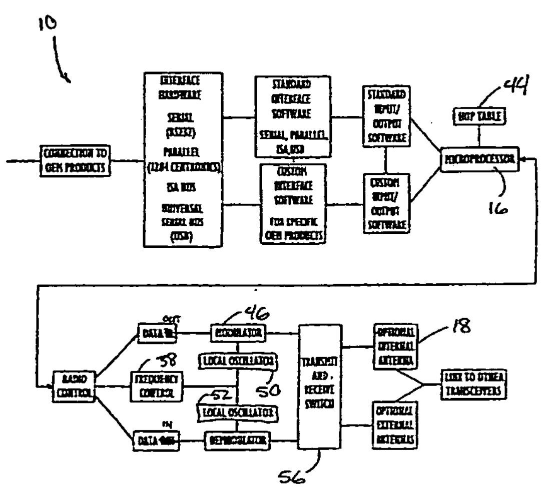
(81) Designated States (*national*): AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CR, CU, CZ, DE, DK, DM, DZ, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZW.

(84) Designated States (*regional*): ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published:
— *With international search report.*

[Continued on next page]

(54) Title: STORE AND FORWARD COMMUNICATIONS METHOD FOR DATA TRANSFER BETWEEN SPREAD SPECTRUM, FREQUENCY HOPPING DATA TELEMETRY TRANSCEIVERS



(57) Abstract: The wireless transceivers include RF and computer control components in a compact package approximately the size of a deck of cards and is adapted to be built into original equipment manufacturer (OEM) products to support a wide range of wireless data telemetry applications. Each transceiver includes a shielded RF board or module with a frequency hopping transmitter and receiver (46), an antenna (18), and a digital control board or module. The transceiver functions as a half duplex, bi-directional communication device. The transmit and receive functions are time interleaved in a non-overlapping fashion. The RF Board consists of a transmitter, receiver, frequency synthesizer (38) and T/R switch (56), each controlled by an external microprocessor (16) to either transmit serial data or receive serial data.

WO 01/33796 A1

— *Before the expiration of the time limit for amending the claims and to be republished in the event of receipt of amendments.*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

STORE AND FORWARD COMMUNICATIONS METHOD FOR DATA TRANSFER BETWEEN SPREAD SPECTRUM, FREQUENCY HOPPING DATA TELEMETRY TRANSCEIVERS

BACKGROUND OF THE INVENTION

Related Application Information:

The instant non-provisional patent application claims benefit of copending provisional patent application number 60/163,833, entitled Communication System & Method for Dynamically Establishing and Maintaining a Plurality of Communication Links, filed November 5, 1999, the entire disclosure of which is incorporated herein by reference.

Field of the Invention:

The present invention relates to a store and forward method for collection and transmission of data via wireless data telemetry utilizing a plurality of transceivers.

Discussion of the Prior Art:

Most of the prior art wireless data transmission products utilize standard RF technology, i.e., radios, the same technology used in vehicle dispatch and police communication systems. Standard RF products are relatively simple and inexpensive to build, but for operation FCC licenses may be required. RF transmissions are susceptible to interference from a growing number of sources and to interception by readily available eavesdropping equipment. The unreliable quality of standard RF transmissions makes the technology unsuitable for applications where all of the information transmitted must be accurate, complete, and secure.

In order to overcome the shortcomings of standard RF transmission methods, direct sequence spread spectrum (DSSS) was developed. DSSS radios divide or slice transmissions into small bits, thereby spreading energy from the bits

simultaneously across a wide spectrum of radio frequencies. DSSS is a relatively unreliable transmission medium, however, because spreading the message across a wide spectrum greatly reduces the strength of the radio signal carrying the message on any one frequency. Since a DSSS receiver must simultaneously monitor the entire allotted spectrum, severe interference from a high energy RF source within the monitored spectrum can pose an insurmountable problem. DSSS performance also degrades quickly in shared-service environments having multiple radio systems operating simultaneously.

Frequency hopping spread spectrum (FHSS) technology was developed by the U.S. military to prevent interference with or interception of radio transmissions on the battle field and is employed by the military in situations where reliability and speed are critical. Standard RF and DSSS cannot match the reliability and security provided by frequency hopping. Instead of spreading (and therefore diluting) the signal carrying each bit across an allotted spectrum, as in DSSS, frequency hopping radios concentrate full power into a very narrow spectral width and randomly hop from one frequency to another in a sequence within a defined band, up to several hundred times per second. Each FHSS transmitter and receiver coordinate the hopping sequence by means of an algorithm exchanged and updated by both transmitter and receiver on every hop. Upon encountering interference on a particular frequency, the transmitter and receiver retain the affected data, randomly hop to another point in the spectrum and then continue the transmission. There should always be frequencies somewhere in the spectrum that are free of interference, since neither benign producers of interference or hostile jammers will

likely interfere with all frequencies simultaneously and at high power radiation levels, and so the frequency hopping transmitter and receiver will find frequencies with no interference and complete the transmission. This ability to avoid interference enables FHSS radios to perform more reliably over longer ranges than standard RF or DSSS radios. In the prior art, frequency hopping FHSS communication systems have been used almost exclusively in the extremely expensive robust military or government communication systems.

Generally speaking, data telemetry is the transmission of short packets of information from equipment or sensors to a recorder or central control unit. The data packets are transferred as electric signals via wire, infrared or RF technologies and data is received at a central control unit such as a computer with software for automatically polling and controlling the remote devices. The control unit analyzes, aggregates, archives and distributes the collected data packets to other locations, as desired, via a local area network (LAN) and/or a wide area network (WAN). Wireless data telemetry provides several advantages over data telemetry on wired networks. First, wireless systems are easier and less expensive to install; second, maintenance costs are lower; third, operations can be reconfigured or relocated very quickly without consideration for rerunning wires, and fourth, wireless telemetry offers improved mobility during use.

Not just any wireless telemetry system will do for many applications, however. The realities of the marketplace dictate that data telemetry cannot be the most expensive part of a system having commercial application. For example, if a retail point-of-sale cash register is to be configured with a wireless data telemetry radio;

the radio cannot be more expensive than the cash register. In many commercial applications, buyers have fixed expectations for what things cost and new features, however useful, cannot substantially exceed those expectations. Thus, it would be best if the wireless data telemetry radio were free. In the interest of providing the most economical wireless data telemetry radio, a transceiver with a shared antenna for both transmit and receive segments is suggested, but how is the switching between transmit function and receive function to be accomplished? The off-the-shelf transmit/receive (T/R) switches are expensive, have a high parts count, and are often configured such that the components within the switch dissipate transmitter energy when in the receive state, adding heat and raising the energy required to operate the wireless data telemetry transceiver.

It is desirable to have a wireless data telemetry radio that is small, light, resistant to interference from adjacent RF noise sources, and uses as little energy as possible.

The Federal Communications Commission (FCC) has designated three license-free bandwidth segments of the radio frequency spectrum and made them available for industrial, scientific and medical (ISM) use in the United States. These three segments are 900 MHZ, 2.4 GHz and 5.8 GHz. Anyone may operate a wireless network in a license-free band without site licenses or carrier fees and is subject only to a radiated power restriction (i.e., a maximum of one watt radiated power). The radio signals transmitted must be spread spectrum. Foreign national spectrum regulation organizations and international telecommunications bodies have also agreed to recognize a common license-free ISM frequency at 2.4 GHz, and so

a defacto international standard for license-free ISM communications has emerged. The ISM band at 2.4 GHz provides more than twice the bandwidth capacity and is subject to far less congestion and interference than the ISM band at 900 MHZ. Several industrial nations do not permit a license-free ISM band at 900 MHZ and relatively few nations have a license-free ISM band at 5.8 GHZ, but the United States, Europe, Latin America and many Asian countries have adopted an ISM band at 2.4 GHZ.

One problem with configuring a wire-replacement communications system is that geographic areas covered may be limited by radiated RF power constraints. Additionally, to really exploit the promise of a self configuring data telemetry system, a network of wire replacement radios should be able to configure itself on the fly, so to speak. Otherwise, running wire is just an alternative to programming wire replacement radios.

What is needed, then, is an inexpensive, easy to use and robust data telemetry and communication system including a plurality of inexpensive and compact transceivers, preferably operating in the common license-free ISM frequency band, and providing reliable communications for a variety of users in commercial and industrial environments.

OBJECTS AND SUMMARY OF THE INVENTION

Accordingly, it is a primary object of the present invention to overcome the above mentioned difficulties by providing an economical, compact wireless data telemetry transceiver network adapted to establish and maintain communication links, preferably in the license-free ISM frequency band at 2.4 GHz.

Another object of the present invention is to increase the geographic area which can be covered by a transceiver or node.

Yet another object of the present invention is to implement an economical and reliable self configuring store and forward network using the economical spread spectrum frequency hopping transceivers of the present invention.

Still another object of the present invention is to provide a method for a given transceiver to discover its place in a network of transceivers.

The aforesaid objects are achieved individually and in combination, and it is not intended that the present invention be construed as requiring two or more of the objects to be combined unless expressly required by the claims attached hereto.

In accordance with the present invention, an economical, compact wireless data telemetry transceiver is adapted to establish and maintain communication links at 2.4 GHz in the license-free ISM frequency band and, in a preferred embodiment, provides the optimum balance between data rate and range, providing, 9.6 kilobits per second (9.6 Kbps) data transmission over an outdoor line of sight range of approximately 35 thousand meters. In an alternate embodiment, designed to comply with European (EPO) regulations, the through the air data rate is raised to 250 Kbps, providing 38.4 Kbps of serial baseband data in a full duplex mode over a reduced line of sight range.

The communication system of the present invention includes components ideally suited to specific wireless data telemetry applications. A transceiver is configured as a printed circuit card having an edge connector. The wireless transceiver includes RF and computer control components in a compact package

approximately the size of a deck of cards and is adapted to be built into original equipment manufacturer (OEM) products to support a wide range of wireless data telemetry applications. Each transceiver includes a shielded RF board or module with a frequency hopping transmitter and receiver, an antenna, and a digital control board or module. The digital control module performs RF module and application interface management and an application interface is included to communicate with specific OEM products utilizing serial transistor/ transistor logic (TTL) or other standard interfaces. The transceiver operates in the license-free portion of the FCC designated ISM frequency band at 2.4 GHz; the transceiver transmits and receives data at 9.6 Kbps at ranges of up to 1500 feet when used indoors with the integrally housed antenna, or up to 12 miles line of sight when used outdoors with an optional directional antenna. The transceiver transmits or receives on any of 550 independent, non-interfering frequencies. When using the transceiver, a data telemetry network can readily be configured for either point-to-point (e.g. wire replacement) or host-to-multipoint networks linked to a user's existing computer or to telephone networks via a system gateway. Optionally, up to 5 collocated independent networks may operate simultaneously, and data security is provided by rapid and random frequency changes (i.e., frequency hopping); the transceiver can optionally be used with data encryption software for providing secure, coded transmissions.

Alternatively, a connector transceiver can be attached to a computer or other device using a standard serial (RS232) port. The connector duplicates the functions of the transceiver but is housed in an enclosure having a cord terminated with an

RS232 compatible connector. The connector can therefore be used with a wide variety of existing products such as cash registers, ATM machines, laptop computers or any other computer controlled device having an RS232 port.

The transceiver functions as a half duplex, bi-directional communication device; preferably, transmit and receive functions are time interleaved in a non-overlapping fashion, consistent with the requirements of a frequency hopping radio. The transmit interval is restricted to less than 0.4 seconds. In the course of a normal information exchange, a given transmission is generated on a frequency selected from a set of all available hop frequencies. The transmission is limited in duration to the availability of incoming data, and following the transmission, the radio switches to a receive mode and processes any incoming data. Once reception is complete, the transmit interval/receive interval cycle is restarted on a new frequency selected from the hop frequency set. Transmit receive cycling continues until all 75 unique frequencies in the set have been used, whereupon the frequency selection process reenters the top of the table and begins reusing the same 75 frequencies.

Transmitted data is directly modulated onto a synthesized carrier by use of minimum shift keying (MSK) modulation. The receiver is a dual conversion super heterodyne, down converting the received signal first to a 315 MHZ intermediate frequency (IF) signal and then down converting a second time to a 10.7 MHZ IF signal. Demodulation is accomplished using a limiter/discriminator circuit and the demodulated data is recovered from the demodulator output by processing through a comparator. First and second local oscillators (LOs) are controlled in frequency by use of a single loop indirect frequency synthesis. Samples of both first and second

voltage controlled oscillators (VCOs) are divided down using phase-locked loop integrated circuit elements, where each sample is compared to an onboard 8 MHZ crystal reference oscillator. During the transmit interval, a single transmitter VCO is controlled by the same device and in the same manner. To minimize total power consumption within the transceiver, portions of circuitry not in use during either the transmit or receive intervals are disabled under control of the system controller.

The RF Board consists of a transmitter, receiver, frequency synthesizer and T/R Switch. Each of these sections is controlled by an external microprocessor to either transmit serial data or receive serial data. The basic transmitted signal is generated by a voltage-controlled-oscillator (VCO) that operates in the 2.4 to 2.4835 GHZ frequency band. The signal is then amplified by three stages of amplification. All three amplification stages and the VCO are switched ON for transmit and switched OFF for receive. A power amplifier stage provides 26 dBm of output power to drive the antenna. This stage also uses a GaAs RF Power FET and a similar power control circuit. The transmitted signal passes through the T/R switch and a 2.44 GHZ 4-pole bandpass filter to the antenna. Both the T/R switch and the bandpass filter are implemented using strip line on a separate daughter board.

The receiver section uses dual conversion with a first IF of 315 MHZ and a second IF of 10.7 MHZ. The received signal from the antenna passes through the same 2.44 GHZ filter the transmitted signal passed through and then passes through the T/R switch to the receiver.

The analog serial data stream is digitized by thresholding the signal using a comparator and a threshold generated from a peak follower. The peak follower

follows both the positive and negative peaks of the analog serial data stream and then generates a threshold signal that is half way between the two peaks. The output of the comparator is the digital received signal output to the digital board.

The RF Board includes an I/O Interface which consists of two mechanical connections. Most of the connections are made via a 20 pin dual in-line header. The other connection is for the antenna and is a microstrip pad and ground connection to which the coaxial antenna cable is soldered. TTL-compatible input signals on the Rx/Tx- pin are used to control the T/R switch. A logic high on this pin puts the T/R switch in the receive position and a logic low puts it in the transmit position. Before the radio switches from receive mode (Rx) to transmit mode (Tx), the T/R switch should be put in the Tx position. When switching from Tx to Rx the T/R switch should remain in the Tx position until after the radio is switched from Tx to Rx.

The RF Board includes an RF I/O connection. When data is presented to the serial port of the digital board, firmware on the digital board will cause the radio to hop on 75 frequencies in the 2400-2483.5MHz band. The dwell time for each hop is 31.6 ms. During a single hop the carrier is frequency modulated with the transmit serial data stream from the digital board. Immediately after the transmit time period the radio switches to the receive mode.

In accordance with the present invention, a method of implementing a store and forward network protocol using spread spectrum transceivers permits the objects identified above to be achieved. The ability to receive data from a transceiver, store it, and then retransmit it greatly increases the geographic area that

can be covered by an RF transceiver. The use of RF and in particular spread spectrum transmission adds a unique variable to the standard store and forward implementation commonly known.

Usually, an RF transceiver is limited to transferring data with only one other transceiver at a time when in a non-broadcast mode. In a wire line system, data can be flowing inbound from one node while being out put to another node, but this is not the case in an RF system. An RF system is also easily affected by changes to the environment, making temporary node outages more prevalent than in a wire line system. A number of features are unique to the spread spectrum store and forward system of the present invention, including: a method used by a transceiver or node to discover its place in a network, a method by which routing tables are developed and become known to the network, a method by which alternate routing is accomplished, a method by which acknowledgements and sequence numbers are employed to allow the network to know when duplicate or missing frames occur, a method by which data flow control is managed, a method by which data is aggregated from multiple nodes for transmission, and a method for managing concurrency of communications with in the network.

In accordance with the present invention, a system has been developed for transceivers or nodes to discover their place in the network. To be able to construct a useful routing table, it is necessary to know the nodes that any given node can communicate with, and the quality of that communications link. In a spread spectrum system, the quality of a link between nodes can vary with the frequency used to communicate. For example, a link between nodes that is excellent at the

low end of the spectral band may be poor at the upper end of the spectral band. Having the node discover its own position relieves the network designer of the task of determining the connectivity of the nodes and the quality of the connections in the network.

The method of discovery works as follows:

A node that has no knowledge of the network will formulate a "clear node request" message. The node will set the destination address of the message to the broadcast address and transmit the request. The receiving node will remove all knowledge of this node from its database of nodes it can communicate with. This step assures that a node that has been replaced or repositioned in the network will be seen as new by those nodes that can hear it. The message is transmitted several times over a predetermined time period to increase the likelihood of the message being received.

The node will next formulate a "join network request" message. The node will set the destination address of the message to the broadcast address and transmit the request. The receiving nodes will create an entry in their database of adjacent nodes for this node. Along with the node ID, the power at which the message was transmitted, the signal strength at which the message was received, and the frequency at which it was transmitted are saved in the database. This message is transmitted several times to increase the likelihood of its receipt.

Each receiving node formulates a join network response that is addressed to the originator of the request. The response is used by the originator of the join network request message to create a database of nodes the receiving node can

communicate with, and, in addition to saving the information about the signal strength, one field of this message tells the originator if the responding node has a path to the host and the shortest length of any paths to the host. If a responder was the host, then the node has a direct path to the host with a path length of one.

Getting knowledge of the network back to the host can occur in one of two ways; either unsolicited by the individual nodes or by the host computer requesting information. Individual nodes send a node registration request to the host computer if they know a path to the host. This node registration request message tells the host about the presence of the node and what it knows of its place in the network. Alternately, the host computer can control the over-the-air traffic by requesting the information itself. The host computer first requests information from the nodes with which it has direct communications. From the data returned, the host then determines the next tier and recursively requests information until all nodes have been contacted. In general, it is better, from an RF communications standpoint, for the host computer to control the process when bringing up a network, if a node is being added or replaced in an existing network, it can originate from the node.

It is not a problem if fewer than all nodes hear all messages, because the host computer resolves incomplete information from the information it does receive. If there is only one node, the host communicates with the node to obtain information as needed, allowing the host to query for any necessary information in the future.

Turning now to Routing Table development, once the host computer has collected all of the node data, it can create a routing table. The routes are weighted

to find the best routes. The largest weighting factor is the length of the route, the shorter the better. The next weighting factor is the quality of the communications along a route. The quality is determined from the power at which the message was transmitted, the signal strength at which the message was received, and the frequency at which it was transmitted at each hop along the path. A final weighting is given to the number of routes a node appears in. This is done to reduce bottle necks which can occur when a node is shared by multiple routes. The individual weighting factors are aggregated to create a single cumulative weighting for a route.

From these weighted routes the host computer will generate a routing table with a primary and secondary route to each node. To reduce the memory required to store routing information at the individual nodes, and to decrease the amount of data that must be transmitted to disseminate the routing information, the host computer will produce a vector table for each node. At a minimum the vector table instructs the node whether it has a path to the destination and if it does, the intermediate node to be used when transmitting to that destination. The vector table also includes a time to live expressed as a maximum number of hops for delivery. The host then disseminates the vector tables by directly sending them to appropriate nodes. When a node has data to transfer to a particular destination, and it is not the originator of the data, it looks for that destination in its vector table. If a route exists to the destination the intermediate node ID is taken from the table and placed in the header of the data frame. A "time to live" metric in the header is decremented and, if it has not gone to zero, the frame is passed on to the next intermediate. If the time to live is zero, the frame is dropped and the event is recorded for diagnostic

purposes. If a node originates data for transmission, it performs the same operation described above but it must also fill in the header with the time to live from the vector table.

Preferably, Primary & Secondary routes are stored. The primary route is tried first. If a node has information, before sending any data from a specific originator to a specific destination, that the primary route is not available, then the secondary route is chosen. Once data has been sent using either the primary or secondary routes, changing routes can cause uncertainties in the delivery of the frame. For example, duplicate frames may be received by the final destination (one from the primary route and one from the secondary route). The status of the hops in a route are monitored by the host computer. When the host computer has information on a problem, it can notify the node adjacent to the bad hop of the problem; here, "hop" means an interval and event for transmission of data between two nodes on a network.

Turning now to acknowledgments and sequence numbers, in networking, two types of acknowledgments exist; point-to-point and end-to-end. The point-to-point acknowledgment is used by a receiver to inform the transmitter that the data has been successfully received. The end-to-end acknowledgment is used to inform the originator that the data has reached the final destination. A point-to-point acknowledgment does not inform the originator about whether the data has reached its final destination.

Whether to use either or both types of acknowledgment depends on three things;

if a 'best effort' or guaranteed delivery is required, if the network nodes have the processing power and memory required to perform packet assembly, and if the end-point equipment has the processing power and memory required to perform packet assembly.

Best effort delivery means that the communications protocol will attempt delivery but will not guarantee it. This type of delivery is used when there is no requirement that all data reach the final destination, such as for a system where periodic status is sent and it is not important that all status messages are received, and when the network resources can not support the over head of guaranteed delivery.

As used here, guaranteed delivery means that the system can detect if the data has been delivered and will continue, with in reasonable limits, to attempt delivery until notification that the data has arrived is received. Guaranteed delivery can be implemented both point-to-point or end-to-end. Since delivery is guaranteed, data frames must be kept until their receipt has been acknowledged.

It is possible to use a combination of best effort and guaranteed delivery in the same network and use of these strategies can change based on the quality of service associated with a particular data stream.

Other features are also possible; with data aggregation, as information moves up the chain to the Host, or filters down to end nodes, it is aggregated. With network concurrency, multiple 'independent' nodes are able to exchange data within the overall network.

The above and still further objects, features and advantages of the present invention will become apparent upon consideration of the following detailed description of a specific embodiment thereof, particularly when taken in conjunction with the accompanying drawings, wherein like reference numerals in the various figures are utilized to designate like components.

BRIEF DESCRIPTION OF THE DRAWINGS

Fig. 1 is a block diagram of a frequency hopping spread spectrum transceiver, in accordance with the present invention.

Fig. 2 is a perspective view of the transceiver of Fig. 1, in accordance with the present invention.

Fig. 3 is a perspective view of a long range connector, in accordance with the present invention.

Fig. 4a is a perspective view of a high gain directional antenna adapted for use with the transceiver of Fig. 1, in accordance with the present invention.

Fig. 4b is a perspective view of an omni-directional antenna adapted for use with the transceiver of Fig. 1, in accordance with the present invention.

Fig. 4c is a perspective view of a larger omni-directional antenna adapted for use with the transceiver of Fig. 1, in accordance with the present invention.

Fig. 4d is a perspective view of a directional antenna adapted for use with the transceiver of Fig. 1, in accordance with the present invention.

Fig. 5 is a diagram illustrating store and forward message propagation, in accordance with the present invention.

Fig. 6 is a diagram illustrating waiting for data propagation in a store and forward network, in accordance with the present invention.

Fig. 7 is a diagram illustrating multi-path data transfer is a store and forward network , in accordance with the present invention.

Fig. 8 is a diagram illustrating local acknowledgment in a store and forward network , in accordance with the present invention.

Fig. 9 is a diagram illustrating local acknowledgments and an end to end acknowledgment in a store and forward network, in accordance with the present invention.

Fig. 10 is a black diagram illustrating the initial operational modes on transceiver boot-up in a store and forward network, in accordance with the present invention.

Fig. 11 is a table describing administrative frames for the discovery period in a store and forward network, in accordance with the present invention.

Fig. 12 is a diagram illustrating the first phase of discovery in a store and forward network, in accordance with the present invention.

Fig. 13 is a diagram illustrating a host joining a store and forward network, in accordance with the present invention.

Fig. 14 illustrates an exemplary store and forward network having a host and nodes A through G arrayed in first and second layers, in accordance with the present invention.

Fig. 15 illustrates an alternative store and forward network in diagramatic form with figure of merit rankings for links between adjacent modes, in accordance with the present invention.

Fig. 16 is a vector table for Node 1 of Fig. 15, in accordance with the present invention.

Fig. 17 is a vector table for Node 2 of Fig. 15, in accordance with the present invention.

Fig. 18 is a vector table for Node 3 of the network of Fig. 15, in accordance with the present invention.

Fig. 19 is a vector table for Node 4 of the store and forward network of Fig. 15, in accordance with the present invention.

Fig. 20 is a vector table for Node 5 of the network of Fig. 15, in accordance with the present invention.

Fig. 21 is a vector table for Node 6 of the network of Fig. 15, in accordance with the present invention.

Fig. 22 is a vector table for Node 7 of the network of Fig. 15.

Fig. 23 is a vector table for Node 8 of the network of Fig. 15.

Fig. 24 is a header data format table for a store and forward network, in accordance with the present invention.

Fig. 25 is a diagram illustrating servicing a priority message in a store and forward network, in accordance with the present invention.

Fig. 26 is a diagram illustrating decrementing the hop counter in a store and forward network, in accordance with the present invention.

Fig. 27 is a diagram illustrating a single-part data transfer in a store and forward network, in accordance with the present invention.

Fig. 28 is a diagram illustrating a multi-path origination in a store and forward network, in accordance with the present invention.

Fig. 29 is a diagram illustrating multi-path timing in a store and forward network, in accordance with the present invention.

Fig. 30 is a diagram illustrating multi-source transfers in a store and forward network, in accordance with the present invention.

Fig. 31 is a diagram illustrating a network fragment for multi-source transfer with step numbers in a store and forward network, in accordance with the present invention.

Fig. 32 is a diagram illustrating data error recovery in a store and forward network, in accordance with the present invention.

Fig. 33 is a diagram illustrating an undeliverable data error sequence in a store and forward network, in accordance with the present invention.

Fig. 34 is a diagram illustrating a sequence having beginning frames lost in a store and forward network, in accordance with the present invention.

Fig. 35 is a diagram illustrating a sequence with lost ending frames in a store and forward network, in accordance with the present invention.

Fig. 36 is a diagram illustrating a sequence with lost intermediate frames in a store and forward network, in accordance with the present invention.

Fig. 37 is a diagram illustrating a serial input buffer used in a store and forward network, in accordance with the present invention.

Fig. 38 is a diagram illustrating data flow between buffers in a store and forward network, in accordance with the present invention.

Fig. 39 is a table illustrating administrative frame formats in a store and forward network, in accordance with the present invention.

Fig. 40 is a diagram illustrating fault management for a host to multipoint configuration in a store and forward network, in accordance with the present invention.

Fig. 41 is a ZNET network view of a store and forward network, in accordance with the present invention.

DESCRIPTION OF THE PREFERRED EMBODIMENT

In accordance with the present invention, a network comprises a plurality of frequency hopping spread spectrum communication transceivers 10, each adapted to dynamically establish and maintain communication links and including components ideally suited to wireless data telemetry applications. As shown in Figs 1 and 2, transceiver 10 is configured as a stacked pair of printed circuit cards including a digital board 12 connected to a shielded RF board 13, the digital board carries a multi-pin connector 14. Transceiver 10 includes RF and computer control components in a compact package approximately the size of a deck of cards and is adapted to be built into original equipment manufacturer (OEM) products to support a wide range of wireless data telemetry applications. Each long range transceiver 10 includes a shielded RF board or module with a frequency hopping transmitter and receiver, an antenna, and a digital control board or module. The digital control module micro processing unit (MPU) 16 performs RF module and application

interface management and an application interface is included to communicate with specific OEM products utilizing serial (transistor/transistor logic, TTL) or other standard interfaces. Transceiver 10 operates in the license-free portion of the FCC designated ISM frequency band at 2.4 GHZ, transmitting and receiving data at 9.6 Kbps at ranges of up to 1500 feet when used indoors with the integrally housed antenna 18, or up to 12 miles line of sight when used outdoors with an optional directional antenna.

As noted above, in an alternate embodiment of transceiver 10 designed to comply with European (EPO) regulations, the through the air data rate is raised to 250 Kbps, providing 38.4 Kbps of serial baseband data in a full duplex mode over a reduced line of sight range.

Transceiver 10 transmits or receives on any of 550 independent, non-interfering frequencies. When using transceiver 10, a data telemetry network can readily be configured for either point-to-point (e.g. wire replacement) or host-to-multipoint networks linked to a user's existing computer or to telephone networks via a system gateway. Optionally, up to 5 collocated independent networks may operate simultaneously, and data security is provided by rapid and random frequency changes (i.e., frequency hopping); transceiver 10 can optionally be used with data encryption software for providing secure, coded transmissions.

Alternatively, a long range connector transceiver 20 as shown in Fig. 3 can be attached to a computer or other device using a standard serial (RS232) port. The long range connector 20 duplicates the functions of the long range transceiver of Figs. 1 and 2 but is housed in an enclosure 22 having an RS232 compatible

connector 26. The long range connector 20 can therefore be used with a wide variety of existing products such as cash registers, ATM machines, laptop computers or any other computer controlled device having an RS232 port and capable of utilizing a frequency hopping spread spectrum communication system software package used to configure a user's or vendor's particular system.

As best seen in Figs 4a-4d, a plurality of optional antennas can be used with either transceiver 10 of Fig. 2 or the long range connector 20 of Fig. 3. In particular, the four inch high mast antenna 30 of Fig. 4b provides moderately enhanced performance and an omnidirectional pattern; the 28 inch high phased array antenna 32 of Fig. 4c provides substantially improved performance in all horizontal directions. The 6 inch flat square panel antenna 34 of Fig. 4d provides substantially improved performance in a single direction, and the 30 inch long tube antenna 36 of Fig. 4a provides dramatically improved performance in a single direction by providing a highly directional beam width. The standard antenna 18 included with either the long range connector 20 of Fig. 3 or the long range transceiver 10 of Fig. 2 is an omni-directional antenna having vertical polarization and a spherical radiation pattern. Standard antenna 18 is built into transceiver 10 or connector housing 22 and does not require an added cable. The four optional antennas of Figs 4a-4d are adapted to be connected using selected cable links or connectors, as required for a specific application.

Transceiver 10 functions as a half duplex, bi-directional communication device over the air. The transmit and receive functions are time interleaved in a non-overlapping fashion, consistent with the requirements of a frequency hopping

radio. The transmit interval is restricted to less than 0.4 seconds on any particular frequency within a thirty second interval. In the course of a normal information exchange, a given transmission is generated on a frequency selected from a set of all available hop frequencies stored in hop table 44. The transmission is limited in duration to the availability of incoming data (or the data payload size for that frame) and following the transmission, the radio switches to a receive mode and processes any incoming data. Once reception is complete, the transmit interval/receive interval cycle is restarted on a new frequency selected from the hop frequency set. Transmit receive cycling continues until all 75 unique frequencies in the set have been used, whereupon the frequency selection process reenters the top of the hop table and begins reusing the same 75 frequencies.

Transmitted data is directly modulated using modular 46 onto a synthesized carrier by use of minimum shift keying (MSK) modulation. The receiver is a dual conversion super heterodyne, down converting the received signal first to a 315 MHZ intermediate frequency (IF) signal and then down converting a second time to a 10.7 MHZ IF signal. Demodulation is accomplished using a limiter/discriminator circuit and the demodulated data is recovered from the demodulator output by processing through a comparator. First and second local oscillators (LOs) 50,52 are controlled in frequency by frequency control circuit 38 which performs a single loop indirect frequency synthesis. Samples of both first and second voltage controlled local oscillators (VCOs) 50,52 are divided down using phase-locked loop integrated circuit elements, where each sample is compared to an onboard 8 MHZ crystal

reference oscillator. During the transmit interval, a single transmitter VCO is controlled by the same device and in the same manner.

To minimize total power consumption within the transceiver, portions of circuitry not in use during either the transmit or receive intervals are disabled under control of the system controller 16.

Frequency management is accomplished by a method incorporated in the transceiver control software. The transceiver initially powers up in an "idle slave" mode and operates in receive mode only, stepping through all 75 hop frequencies while "listening" for an incoming header packet matching the idle slave's local address.

When data is presented to a transceiver via its local communications port (e.g., RS-232), the transceiver immediately shifts from idle slave mode to a "master search" mode wherein the master transmits and then listens for (receives) an acknowledgment signal from a targeted remote slave device (i.e., a transceiver in idle slave mode). The transmit and receive periods each represent one-half of a complete hop interval. The master continues to search for the slave device until a valid acknowledgment is received or until a predetermined time-out period expires. The initiation of master search mode starts at whichever hop frequency the transceiver was previously using while in idle slave mode and continues to step through the hop table selecting frequencies in turn. Since the incoming data is a synchronous in nature, the master transceiver essentially begins this process at a random point within the hop table.

An idle slave device, after receiving a valid header data packet, transmits an acknowledgment packet during the master's listening phase of the hop interval, thereby creating a synchronized and linked session for data transfer. Once linked, the master and slave transceivers increment through all 75 entries in the hop table for as long as incoming data is present for either unit, after a programmable time-out period. The master transmits during the first half of each hop interval and the slave transmits during the second half of the interval with the slave device adjusting its response time in accordance with the received data packet, thereby maintaining synchronization between both master and slave devices. When neither master nor slave has any additional data to transmit, both units return to the idle slave mode after a preprogrammed time-out period.

The receiver portion of the transceiver is implemented very economically; the recovered analog serial data stream is digitized by thresholding the signal using a comparator and a threshold generated from a peak follower. The peak follower follows both the positive and negative peaks of the analog serial data stream and then generates a threshold signal that is half way between the two peaks. The output of the comparator is the digital received signal directed to digital board 12. A universal asynchronous receiver-transmitter (UART) is incorporated in each transceiver to process both transmit and received data.

Transceivers communicate using an On-Air Protocol that is stored in firmware and includes specific characteristics for the two types of on-air "frames", i.e., the linking frame and the data frame. The linking frame is transmitted when transceivers are not currently communicating to synchronize them to the same

frequency. Once the transceivers are synchronized, data frames are transmitted until the (then) current session ends, even if there is no data to be sent. "Synchronization", as used here, does not mean that precisely synchronized clocks (i.e., between transceivers) are required, however.

Turning now to a more detailed description of transceiver RF components, RF Board 13 consists of a transmitter, receiver, frequency synthesizer and a transmit/receive (T/R) Switch 56. Each of these sections is controlled by microprocessor 16 to either transmit serial data or receive serial data.

The basic transmitted signal is generated by a voltage-controlled-oscillator (VCO) that operates in the 2.4 to 2.4835 GHZ frequency band. The signal is then amplified by three stages of amplification. All three amplification stages and the VCO are switched ON for transmit and switched OFF for receive.

The first stage of amplification is provided by a bipolar transistor capable of generating at least 10 dBm output power to boost the signal generated by the VCO and drive the exciter stage and to provide some isolation between the power stages and the VCO. The base bias on both the VCO and bipolar amplifier is controlled to provide the transmit ON/OFF function.

The exciter stage boosts the power to at least 22 dBm to drive the power amplifier stage. The is accomplished using a GaAs RF Power FET. A power control circuit is used to generate the gate bias voltage. The circuit is a closed loop control circuit that controls the level of drain current. Different drain current settings are used to control the output power of the amplifier. This includes the OFF state for receive as well as three other power levels. The power level settings are

programmed via two control lines accessible at the RF Board connector. The circuit also controls the turn-on and turn-off times so that spectral splatter can be reduced. The power amplifier stage provides 26 dBm of output power to drive the antenna. This stage also uses a GaAs RF Power FET and a similar power control circuit. The same two control lines that control the exciter power level also control the power amplifier power level. The transmitted signal passes through T/R switch 56 and a 2.44 GHZ 4-pole bandpass filter to the antenna. Both T/R switch 56 and the bandpass filter are implemented using strip line on a separate daughter board.

As noted above, the receiver uses dual conversion with a first IF of 315 MHZ and a second IF of 10.7 MHZ. The received signal from the antenna passes through the same 2.44 GHZ filter the transmitted signal passed through and then passes through the T/R switch 56 to a low noise amplifier (LNA). The filter acts as a preselector to prevent strong out-of-band signals from desensitizing the receiver. The LNA provides approximately 15 dB gain with 2 dB noise figure. An image rejection filter centered on 2.44 GHZ follows the LNA, and is implemented as a strip-line 2-pole bandpass interdigital filter on a separate daughter board.

The first hetrodyne mixer is after the image filter. The local oscillator (LO) for the first mixer is a 2.085 to 2.1685 GHZ VCO which is part of the synthesizer. At each hop frequency, the first LO is tuned to a frequency 315 MHZ below the receive frequency. The LO signal passes through the LO filter to the first mixer. This filter is also implemented on the daughter board using strip line and is a 2-pole bandpass interdigital filter centered at 2.125 GHZ. The output of the mixer consists of a number of signals, one of which corresponds to the first IF of 315 MHZ. A 315 MHZ

surface acoustic wave (SAW) filter follows the mixer to select the first IF from amongst the products of the mixer. Following the SAW filter is a stage of 315 MHZ amplification. The signal then passes to the second hetrodyne mixer. The second mixer uses a high side LO frequency of 325.7 MHZ so that mixing products are not generated on other channels in the 2.4 to 2.4835 GHZ frequency band. The desired result of this mixer is a 10.7 MHZ signal which then passes through a 10.7 MHZ ceramic 150KHz bandpass filter to an IF amplifier. The signal passes through another 10.7 MHZ ceramic 150 KHz bandpass filter after the IF amplifier and then to the limiter amplifier. Both of these amplifiers and the active part of the discriminator are a part of an IF processing chip. A third 10.7 MHZ ceramic 400 KHz bandpass filter is used as the delay element in the discriminator. The discriminator produces an analog version of the serial data stream.

The analog serial data stream is digitized by thresholding the signal using a comparator and a threshold generated from a peak follower. The peak follower follows both the positive and negative peaks of the analog serial data stream and then generates a threshold signal that is half way between the two peaks. The output of the comparator is the digital received signal output to the digital board.

The frequency synthesizer generates the modulated transmit signal, the receiver first LO, and the receiver 2[nd] LO, each phase locked to the on-board 8 MHZ reference.

The 8 MHZ reference is a crystal oscillator that is controlled by the off-board microprocessor 16. To enable a cost effective solution for the reference an inexpensive crystal is utilized. Because a frequency tolerance of 3 parts per million

(ppm) must be maintained for the transceiver to communicate, a frequency compensation routine is programmed for execution with microprocessor 16. The compensation deals with both the initial crystal manufacturing tolerance and maintaining tolerance over the specified –20 to 70 degrees Celsius temperature range.

The transmitted signal is generated by a VCO, switched on during transmit, operating over a 350 MHZ tuning range roughly centered on 2.44 GHZ. During operation the VCO only tunes in the 2.4 to 2.4835 GHZ band. Having a larger tuning range allows for manufacturing tolerances without the need to tune the oscillators for each manufactured board. During operation, the synthesizer chip is programmed to the required hop frequencies. The chip has a fast and a slow loop mode. When a frequency is first programmed the chip is placed in the fast mode. After a selected interval of approximately 3 ms the chip is switched to slow mode. This allows the tuning loop time to settle on the correct frequency and then slows the loop so that frequency modulation of the transmitted signal by the data can be accomplished by impressing very small changes on the tuning voltage. If the tuning loop were not slowed then it would be able to partially correct the small tuning voltage impressions and cause pulse droop on the subsequently received signal.

The first LO signal is generated by second LO VCO 52, switched on during receive in the place of the transmit VCO 50. This receive VCO 52 shares the same connections to the synthesizer chip that the transmit VCO 52 does. As with the transmit VCO it has a tuning range of 350 MHZ to allow for manufacturing tolerances. Its tuning range is roughly centered on 2.125 GHZ which is 315 MHZ

below the transmit frequencies. During operation, it hops to frequencies in the 2.085 to 2.1685 GHZ band. Unlike the transmit VCO, the synthesizer chip is tuned to a frequency in fast mode and never switched to slow mode. This allows the synthesizer combination to have a much better close-in phase noise.

The second LO signal is generated by a VCO that has approximately a 35 MHZ tuning range centered on 325.7 MHZ. This VCO is connected to the low frequency section of the dual frequency synthesizer chip. This VCO and this section of the synthesizer chip are energized only while receiving. It is always programmed to 325.7 MHZ.

The RF Board I/O Interface consists of two mechanical connections. Most of the connections are made via a 20 pin dual in-line header. The antenna connection is a microstrip pad and ground to which the coaxial antenna cable is soldered. TTL-compatible input signals on an Rx/Tx- pin are used to control the Rx/Tx switch 56. A logic high on this pin puts the Rx/Tx switch 56 in the receive position and a logic low puts it in the transmit position. Before the radio switches from Rx mode to Tx Mode the Rx/Tx switch 56 should be put in the Tx position. When switching from Tx mode to Rx mode the switch 56 should remain in the Tx position until after the radio is switched from Tx to Rx.

Turning now to the method of implementing a store and forward network protocol using spread spectrum transceivers, the ability to receive data from a transceiver 10, store it, and then retransmit it greatly increases the geographic area that can be covered by an RF transceiver or node. The use of RF and in particular

spread spectrum transmission adds a unique variable to the standard store and forward implementation commonly known.

Usually, an RF system is easily affected by changes to the environment, making temporary node outages more prevalent than in a wire line system. A number of features are unique to the spread spectrum store and forward system of the present invention, including: a method used by a transceiver or node to discover its place in a network, a method by which routing tables are developed and become known to the network, a method by which alternate routing is accomplished, a method by which acknowledgments and sequence numbers are employed to allow the network to know when duplicate or missing frames occur, a method by which data flow control is managed, a method by which data is aggregated from multiple nodes for transmission, and a method for managing concurrency of communications with in the network.

In accordance with the present invention, a system has been developed for transceivers or nodes to discover their place in the network. To be able to construct a useful routing table, it is necessary to know the nodes that any given node can communicate with, and the quality of that communications link. In a spread spectrum system, the quality of a link between nodes can vary with the frequency used to communicate. For example, a link between nodes that is excellent at the low end of the spectral band may be poor at the upper end of the spectral band. Having the node discover its own position relieves the network designer of the task of determining the connectivity of the nodes and the quality of the connections in the network.

The method of network configuration discovery works as follows:

1) A node that has no knowledge of the network configuration will formulate a "clear node request" message. The node will set the destination address of the message to the broadcast address and transmit the request. Each receiving node will remove all knowledge of this node from its database of nodes it can communicate with. This step assures that a node that has been replaced or repositioned in the network will be seen as new by those nodes that can hear it. The message is transmitted several times over a predetermined time period to increase the likelihood of the message being received.

2) The node will next formulate a "join network request" message. The node will set the destination address of the message to the broadcast address and transmit the request. The receiving nodes will create an entry in their database of adjacent nodes for this node. Along with the node ID, the power at which the message was transmitted, the signal strength at which the message was received, and the frequency at which it was transmitted are saved in the database. This message is transmitted several times to increase the likelihood of its receipt.

3) Each receiving node formulates a join network response that is addressed to the originator of the request. The response is used by the originator of the join network request message to create a database of nodes the receiving node can communicate with, and, in addition to saving the information about the signal strength, one field of this message tells the originator if the responding node has a path to the host and the shortest length of any paths to the host. If a responder was the host, then the node has a direct path to the host with a path length of one.

4) Getting knowledge of the network back to the host can occur in one of two ways; either unsolicited by the individual nodes or by the host computer requesting information. Individual nodes send a node registration request to the host computer if they know a path to the host. This node registration request message tells the host about the presence of the node and what it knows of its place in the network. Alternately, the host computer can control the over-the-air traffic by requesting the information itself. The host computer first requests information from the nodes with which it has direct communications. From the data returned, the host then determines the next tier and recursively requests information until all nodes have been contacted. In general, it is better, from an RF communications standpoint, for the host computer to control the process when bringing up a network, if a node is being added or replaced in an existing network, it can originate from the node.

It is not a problem if fewer than all nodes hear all messages, because the host computer resolves incomplete information from the information it does receive. If there is only one node, the host communicates with the node to obtain information as needed, allowing the host to query for any necessary information in the future.

Turning now to Routing Table development, once the host computer has collected all of the node data, it can create a routing table. The routes are weighted to find the best routes. The largest weighting factor is the length of the route, the shorter the better. The next weighting factor is the quality of the communications along a route. The quality is determined from the power at which the message was transmitted, the signal strength at which the message was received, and the

frequency at which it was transmitted at each hop along the path. A final weighting is given to the number of routes a node appears in. This is done to reduce bottle necks which can occur when a node is shared by multiple routes. The individual weighting factors are aggregated to create a single cumulative weighting for a route.

From these weighted routes the host computer will generate a routing table with a primary and secondary route to each node. To reduce the memory required to store routing information at the individual nodes, and to decrease the amount of data that must be transmitted to disseminate the routing information, the host computer will produce a vector table for each node. At a minimum the vector table instructs the node whether it has a path to the destination and if it does, the intermediate node to be used when transmitting to that destination. The vector table also includes a time to live expressed as a maximum number of hops for delivery. The host then disseminates the vector tables by directly sending them to appropriate nodes. When a node has data to transfer to a particular destination, and it is not the originator of the data, it looks for that destination in its vector table. If a route exists to the destination the intermediate node ID is taken from the table and placed in the header of the data frame. A "time to live" metric in the header is decremented and, if it has not gone to zero, the frame is passed on to the next intermediate. If the time to live is zero, the frame is dropped and the event is recorded for diagnostic purposes. If a node originates data for transmission, it performs the same operation described above but it must also fill in the header with the time to live from the vector table.

Preferably, Primary & Secondary routes are stored. The primary route is tried first. If a node has information, before sending any data from a specific originator to a specific destination, that the primary route is not available, then the secondary route is chosen. Once data has been sent using either the primary or secondary routes, changing routes can cause uncertainties in the delivery of the frame. For example, duplicate frames may be received by the final destination (one from the primary route and one from the secondary route). The status of the hops in a route are monitored by the host computer. When the host computer has information on a problem, it can notify the node adjacent to the bad hop of the problem; here, hop" means an interval and event for transmission of data between two nodes on a network.

Turning now to acknowledgments and sequence numbers, in networking, two types of acknowledgments exist; point-to-point and end-to-end. The point-to-point acknowledgment is used by a receiver to inform the transmitter that the data has been successfully received. The end-to-end acknowledgment is used to inform the originator that the data has reached the final destination. A point-to-point acknowledgment does not inform the originator about whether the data has reached its final destination.

Whether to use either or both types of acknowledgment depends on three things;

a) if a 'best effort' or guaranteed delivery is required,

b) if the network nodes have the processing power and memory required to perform packet assembly, and

c) if the end-point equipment has the processing power and memory required to perform packet assembly.

As noted above, best effort delivery, in this context, means that the communications protocol will attempt delivery but will not guarantee it. This type of delivery is used when there is no requirement that all data reach the final destination, such as for a system where periodic status is sent and it is not important that all status messages are received, and when the network resources can not support the overhead of guaranteed delivery.

For guaranteed delivery, the system can detect if the data has been delivered and will continue, with in reasonable limits, to attempt delivery until notification that the data has arrived is received. Guaranteed delivery can be implemented both point-to-point or end-to-end. Since delivery is guaranteed, data frames must be kept until their receipt has been acknowledged. It is possible to use a combination of best effort and guaranteed delivery in the same network and use of these strategies can change based on the quality of service associated with a particular data stream.

As will be described in greater detail hereinbelow, data aggregation permits information moving up the chain to the Host to be aggregated, and with network concurrency, multiple 'independent' nodes are able to exchange data within the overall network.

Turning now to a more detailed description of the preferred embodiment of the store and forward method, and referring to Figs 5-42, this section provides an

overview of the store and forward network using the transceiver 10 of the present invention.

As noted above, a repeater network is necessary when the physical lay-out of a network precludes each transceiver or node from having direct access to the host transceiver or node. In this situation, messages from a remote node which does not have direct access to the host node must be routed by a node adjacent to the remote toward the host node. Several nodes may be required to pass the message along before it reaches its destination at the host.

Intermediate nodes examine the destination address, and if it is not the node's own address the node determines the node to route the data to for it to reach the destination.

The configuration of a repeater network is driven by the physical space in which the network is to be deployed, the 'depth' of the network and the 'width' of the network. The depth of a network is the number of nodes between a remote and the host. The width of a network is the number of nodes a transceiver can directly communicate with. Several general rules can be applied in the design of a network, these are:

- The greater the network depth the longer the delivery time.
- The greater the network width the larger the number of potential routes.

Having a large width is a 2-edged sword; it provides greater flexibility in routing but also greater complexity in route management.

Any message which cannot be contained in a single data frame will be divided into multiple frames and reassembled into the original message by the

destination transceiver. Two methods are available for propagating a message across a network, as shown in Figs 5 and 6.

One - All of the frames in a message could be collected by the node at the next higher level and their receipt acknowledged before they are forwarded on to the next layer node. This method keeps all of the frames together, insuring that the entire message is delivered in order. In a wired environment where a node can receive and transmit simultaneously, this would be a very slow method, since no transmission would take place until all of the data was received. Using the frequency hopping environment of the present invention, this is an efficient method since a transmit – receive pair stay synchronized in the hop table until a session has ended. For large amounts of data, i.e., greater than the amount which can be buffered on a transceiver, the session from the sending unit to the receiving unit may need to be paused until the receiver can clear it's buffer by re-transmitting the data.

Two - After an individual frame of data is received by the next higher layer, that node stops receiving and immediately tries to transmit the frame to the next layer node. In the frequency hopping environment of the present invention, this would be a very inefficient method, since a radio link would need to be acquired for each frame.

The difference between methods One and Two above can be thought of as collecting a bucket of data before retransmitting or sending frame-by-frame.

As seen in Fig. 7, multi-path routing allows data to go over one of several paths. In the frequency hopping environment of the present invention, to achieve the highest throughput, a transceiver cannot be idle while waiting for the next

transceiver in the route to retransmit the data.

If transceiver A, in the example above, has a second path to transfer data over, it does not need to wait for transceiver B to retransmit. The multiple paths from transceiver A to the final destination can be either virtual circuits or dynamically allocated routes.

For nomenclature purposes, a "session" is defined as a virtual connection between two points, the originator and the data endpoint, which exists until all data available for transfer has been transferred, acknowledged, and the session terminated by a session going down message. This definition permits many degrees of freedom when applied to a store and forward network having intermediate nodes performing routing. Optimizing a network requires considering a number of questions. For example, does the session exist between the endpoints or between adjacent nodes? How long does a session last? Is it only until the receiver cannot accept any more information? Is it until all data is sent from the transmitter to the receiver? Is it until all of the data arrives at the final destination? Can more than one transmitter share the same session? For example, an intermediate receives data from unit A and cannot get a link to the next node. The intermediate could go into receive during the random stand-off period and get data from a second unit. When it acquires the link to the next node it could transfer both unit's data. Is the concept of a session necessary? Could a connectionless datagram system work?

A session that spans the entire data path from end-point to end-point is called a virtual circuit. A virtual circuit is setup by either having a predetermined route from the source to the final destination or by determining the route at each node as it is

being setup. Once the circuit is setup it is used for the life of the session. This means that while individual links between nodes my go up and down the virtual circuit through which the data must pass remains constant. The advantages of a virtual circuit are that the cost of determining the route is experienced only at the initiation of the session, and the order in which the data is received is the same as it was sent. The disadvantages are that a failure in the route can halt the flow of data, a node may need to wait for the next node in the route to become available before it can send again, and knowledge of the virtual circuit must be maintained at each node just as session information must be maintained in the current code base. The difficulties this presents are many and magnified when distributed across many nodes.

An alternative to a virtual circuit is to use a connectionless protocol and connectionless datagram service. A connectionless protocol does not maintain the idea of a session from node to node. When a link is brought up, it will remain until all of the data is transferred or the link is broken. If the link is broken (this can be caused by RF or power disruption or by flow controlling data), the transceiver with data to send will attempt to restore the link. The other party to the conversation maintains no history that it was in a conversation (except for the frame sequence ID to expect) and is free to link to another transceiver.

The advantages of a connectionless protocol are:

- In an environment where breaks in communications are frequent a connectionless protocol is advantageous. A session oriented protocol would leave a session up between two nodes when communications are interrupted. This can

have the effect of leaving a node in a state of limbo until either the session is re-established or the error recovery system clears the session; also

- the complexity of maintaining sessions for a virtual circuit is not necessary.

The disadvantage is that the data is not necessarily received in the order it was sent, if a link is broken and reestablished.

In the connectionless protocol of the environment of the present invention, packets are be transferred between two nodes until either:

- all of the data has been transferred,

- the transmitting node is signaled to stop sending by the receiving node, for flow control purposes, or

- the link between the nodes is interrupted.

Related to the concept of sessions is packet acknowledgment. As noted above, two types of acknowledgment exist, point-to-point and end-to-end. Point-to-point acknowledgment assures the sending unit that the next layer node, the receiving node, has correctly received the data sent. The originating unit does not know if the data ever made it successfully to its final destination. For example, if node A gets a local acknowledgment from node B for the data it sent it does not have an indication that the data ever made it to the host. This is the typical behavior of a store and forward network.

End-to-end acknowledgment means that the originating unit does not receive a final acknowledgment until the data is successfully received by the final destination. This means that the originating unit must buffer all data sent (in case retransmission is needed) until it is acknowledged by the final destination. Typically,

end-to-end acknowledgment is performed only for applications where delivery must be guaranteed. This is due to the expense of buffering the data until it is acknowledged. Applications for which end-to-end acknowledgment will be required include; security systems, alarms, and where the transceiver is to serve as a wire replacement.

If a session is established between the originator and the final destination, this does not substitute for end-to-end acknowledgment, as in Fig. 9. The purpose of a session is to maintain a connection between two points until all data is sent, not to acknowledge the receipt of data. If data that was locally acknowledged does not reach the final destination, the originating node would not detect this and retransmit. In Fig. 8, transceiver A has information that all of its data has been successfully sent and will send a 'session going down' message. The session between A and B is complete, but not between B and the Host.

Since it is possible that not all data is received before it is retransmitted, and that the path taken from source to final destination is not always the same, packets may be received out of order. This means that the single bit frame ID which was adequate for message passing in a point-to-point or host to multipoint network is no longer sufficient.

The Store and forward network of the present invention incorporates those features which are best suited to frequency hopping transceivers. The network is self discovering and configuring to optimize the issues of network width and depth. The transceivers take advantage of hop table synchronization by transmitting data until

none is left, one of the transceiver's receive buffers fills, or the link is broken.

The transceiver with data to transfer will attempt to reestablish communications if the link has been broken or flow controlled.

Multi-path routing is optional and may be omitted. Communications between transceivers is connectionless, but only one link can be supported by a transceiver at a time. One transceiver will link to any other with out preference to previous links. All acknowledgments are point-to-point unless supported by a higher level protocol in the devices attached to the transceivers.

The transceiver has several modes of operation. These modes of operation are:

1) Firmware loader – to change the firmware stored in flash memory.

2) Command mode – allows changes to be made to the unit's configuration.

3) Discovery – determines a node's place in the network.

4) Store and Forward – the operational mode where messages are passed through the network.

Referring now to Fig. 10, the system boots and initializes itself and will wait up to 40 msec. to receive a command, placing it into one of the administrative modes. If no command is received with in the 40 msec. time frame the unit will continue to one of the operational states depending on its knowledge of the network.

When it is necessary to reconfigure a remote transceiver, two methods can be used, either administrative frames are sent across the network to the transceiver

or the transceiver is directly connected to a PC running the configuration program. When the transceiver is directly attached to a PC the following protocol is used:

a) the configuration program on the PC is started

b) the transceiver which was turned off is now turned on the PC repeatedly sends an 8 character code to the transceiver to instruct it to go into one of the administrative states. These states are:

i) reprogram flash

ii) configure unit

d) the transceiver goes into receive mode and looks for the character string which indicates the mode to which it should transition.

e) if no code is received with in 40 msec., then the transceiver continues with its normal boot cycle

f) at the end of an administrative state, the PC can command the transceiver into another administrative state, have it reset, or allow it to finish booting

Once the system transitions to the operational state it may not be operational as a store and forward transceiver. The transceiver must have knowledge of the network and its place in the network. If the transceiver does not have this knowledge then the transceiver goes into the discovery mode. Once knowledge of the network is available to the transceiver and the network has been enabled the transceiver will transition to the normal store and forward mode of operation.

<u>Bringing up a New Network</u>

When a new Store and Forward network is installed, it must be configured so that nodes in the network know how to communicate with each other. A Store and

Forward network consists of one "host Transceiver" (designated as such by setting its source ID to zero), one or more remote Transceivers (each with a unique source ID of 1 through 240), whose destination address is zero, and who all have the same network ID, vendor ID, and hop table and Host PC (connected to the host Transceiver) that is running a S&F network management utility.

The Store and Forward network is not straightforward to configure, since intermediate nodes perform routing and multiple routes are possible. To avoid burdening the user with the task of figuring out the network topology and configuring each unit with a primary and secondary route, a method of auto-configuration has been developed. Each unit is still configured by the rules listed above, however, once a unit is powered on, a process of route auto-discovery is begun. The steps of this process of auto-discovery are:

1) a node announces its presence in a network,

2) the neighboring nodes acknowledge the new node,

3) the node registers its location with the Host by informing the Host who its neighbors are.

During this period of discovery, the network will process RF administrative frames and not data. Administrative frames are defined as RF messages whose "msg_type" is neither COMMAND nor DATA. Administrative frames are exchanged between Transceivers only; they have no interaction with the ZNET application.

After discovery is complete, the host PC may request information about the new network by issuing a command (message type is set to FRM_MSG_TYPE_COMMAND).

The "Join Network" Request

A node joining a network needs to know how it fits into the network as shown in Fig. 14. Can it talk directly to the Host? Who can it talk to (e.g., its adjacent nodes)? Which of its adjacent nodes has a path to the host Transceiver? What is the best route to the Host?

To determine these things, after a S&F Transceiver is first powered on, the Transceiver sends out an administrative frame seeking any potential communications partners (e.g. adjacent nodes). If responses are received from Transceivers at different addresses, then it belongs to a S&F network where the respondents are neighboring network members. If one of the respondents has an address of zero then the node has a direct path to the Host.

Assuming there is a new Transceiver, (e.g. Transceiver-A) that has been just powered on, the procedure to join a network is as follows:

1) Transceiver-A broadcasts a "Clear-Node" request. Each Transceiver that hears this request will clear the entry for Transceiver-A in its table of adjacent nodes. Transceiver-A may broadcast the Clear-Node request more than once, in order to guarantee that all neighboring Transceivers have a chance to hear the request.

2) After the clear node request process is complete, Transceiver-A broadcasts a "Join-Network" request. Each Transceiver that hears this request will add Transceiver-A's information to its adjacent node table. Transceiver-A may broadcast the Join-Network request more than once in order to guarantee that all neighboring Transceivers have a chance to hear the request. (Note: An adjacent

node will only add Transceiver-A to its adjacent node table, if Transceiver-A is not already in the table.)

3) Each Transceiver that adds Transceiver-A to its adjacent node table will transmit a Join-Network repsonse to Transceiver-A. This indicates to Transceiver-A that the request was received and that the adjacent node now considers Transceiver-A as a neighbor.

4) All Transceivers that have successfully transmitted a Join-Network repsonse to Transceiver-A, will be added to the Transceiver-A's adjacent node table.

Steps 2 through 4 are repeated until the hop table has been traversed twice with out a Join-Network response being received. This is done to insure that each adjacent node has had an opportunity to respond.

It is not imperative that the new remote hears from every available adjacent node. The recursive nature of each node making a Join-Network request greatly increases the likelihood that if Transceiver-A does not hear from Transceiver-B, Transceiver-B will hear from Transceiver-A when it goes through its join cycle. In addition, the Host may have additional information previously collected on the network topology.

<u>The "Join Network" Response</u>

If a neighboring node, Transceiver-B, hears a Join-Network request from Transceiver-A, it checks to see if Transceiver-A is already in its adjacent node table or if this is a new request. If Transceiver-A is not in its table, Transceiver-B responds with a Join-Network response message. This message includes the node ID Transceiver B, the RSSI that was seen for Join-Network request message, the

number of hops to the Host (e.g. the Host path length); a length of 255 means no path, and the quality of service for the respondent's path to the Host (if any).

Transceiver-A will create an adjacent node table from the responses. The adjacent Node ID, RSSI, Transmit Power, Error Count, Host Path Length, Active/Inactive State, Frames Tx count and Frames Rx count.

Where:

Node ID	=	the ID of the adjacent node
RSSI	=	the adjacent node's receive signal strength indicator normalized
Transmit power	=	the power level that the adjacent node transmits with
Error count	=	the number of errors encountered in communicating with the adjacent node.
Host path length	=	the length of any path to the Host (a length of 0 means no path)
Active/Inactive	=	if an adjacent node can no longer communicate yes or no
Frames Tx	=	the number of frames transmitted to this node
Frames Rx	=	the number of frames received from this node

Counters (error count, frames Tx, and Frames Rx) are cleared when they are read. If they accumulate to the maximum value they will remain there until read.

Node Registration

After a Transceiver (e.g. Transceiver-A) has completed its cycle of *Join-Network* request messages, it determines which adjacent node has the shortest route to the Host, and transmits a Node Registration request via that node.

Transceiver-A will periodically repeat the discovery process. If a host Transceiver comes online after Transceiver-A's initial discovery process has completed, Transceiver-A will resend the Node Registration to the Host, once it has discovered the presence of the host Transceiver.

Route Table Generation

The large amount of interconnectivity available in this network makes the task of determining preferred and optional routes large. The quality of each route from a remote to the Host must be measured and the routes compared and ranked. To keep the processing requirements of the TRANSCEIVER to a minimum, the job of creating a route table is delegated to the Host. The Host generates a global route table, which shows the interconnections of the various nodes and the quality of each connection. First, each node is assigned a 'layer' value that is equal to the shortest route from the node to the Host.

Next, to simplify the route table the following rules are employed:

A) The Host must be one end point of a route. See Fig. 13.

B) The shortest route is the best. Routes longer than the three shortest do not need to be considered.

C) If there are multiple routes of the same length, the one with the best quality of service numbers is the very best.

D) If only one route exists it is both the primary and alternate.

E) Routes cannot include the same node more than once.

For example, in Fig. 14, the primary path for node G to the Host, H, would be G – D – H and its alternate would be G – F – C – H. Routes of more than four nodes would not be considered since there are more than two routes of four nodes or less. The routes for node G even after simplification by applying the rules are: G – D – H, G – D – C – H, G – F – C – H, and G – F – B – H.

Quality Of Service

Once the Host computer has calculated the possible routes, it needs to determine the quality of each route. The quality of the routes will be used to assign a figure of merit to each route. The quality of a route from a remote to the Host is measured by examining:

a) the number of nodes in the route,

b) the frame error rate between each node, and

c) the ratio of the time averaged receiver signal strength indicator (RSSI) to the transmission power between each node of the route.

The number of nodes in a route effects the speed with which data is delivered; the smaller the number, the quicker the transmission in general. The frame error rate between nodes is kept as an average over time. In an RF line of sight transmission system many temporary incidents (such as weather, objects moving between the transceivers, etc.) can affect the frame error rate for a limited time, and averaging takes this into consideration. The RSSI is the strength of the RF signal as seen by the receiver. The value of the RSSI in calculating quality of service increases when it is measured against the transmit power level. For example, two RSSI of the same level may not be equal if one unit must transmit at full power and the other can transmit at a low power to achieve this reading. The value used to calculate the quality of service is the ratio of the RSSI to the transmit power level.

The individual quality of service measures are combined to form a figure of merit for each node to node route. The components of the figure of merit are

weighted to give more importance to the length of the route and the least importance to the error rate. The figures of merit of the routes between two endpoints are compared to create a ranking. For example. in Fig. 15, several routes exist from the Host, H, to node 6. The smaller the figure of merit, the better the quality in this example. By taking the sum of the node-to-node links it can be seen that the routes H – 1 – 5 – 6 and H – 4 – 7 – 6 provide the two best and equivalent paths. These routes are not the shortest. Since the length of a route is given the most weight in calculating the best route the shorter routes in this example must have terrible RF characteristics.

Route Table Generation

The individual nodes in a network do not need to know the full route table. Transmitting the full route table to each node would require a large amount of time and memory on the individual transceivers. Instead, the Host computer generates a route table for each node.

The route table is a simple array where the index into the table is the destination node and the elements of the table are a primary and alternate adjacent node to route through to reach that destination. The primary and alternate routes are based on the figure of merit previously calculated. If a path is not available or a node with that ID does not exist, then the route table will contain a 255 for the routing node ID (primary and alternate).

If the data arrived over the RF link and this node is the final destination, then the route table will contain a 254 for the routing node ID. This is a failsafe since the

destination address should previously have been examined and a determination made that this is the final destination.

The number of hops to the destination for each route is also provided so that it can be used as a 'time to live' to safe guard against loops (this will be discussed later).

The Host uses a 'Route Table Update Command' addressed to a specific node to distribute the route tables for each node. The distribution of the route tables provides a benefit beyond allowing nodes to know how to route. If the Host looses its route table it can ask each node for a copy of its route table. From the individual route tables the Host can recreate the route table without all of the overhead of a complete network rebuild.

Orphaned Nodes

It is possible for a node to try and join a network and not be able to find an adjacent node with a route to the Host. This would be the case when building a network from the remotes back to the Host. In this situation, the node is referred to as "orphaned" until it joins the network. If the node remains orphaned for a configurable period of time, it will attempt again to join the network. This will continue until the node is able to join the network.

Operational State

Once the route tables have all been distributed, the network is considered configured. The Host computer will now generate a 'Network In Service' message and send it to all of the nodes in the network individually. At this point the network is considered operational. While the network is operational the Host computer can

update the routing and route tables to reflect changes such as the addition or deletion of nodes. The Host computer can also monitor and manage various aspects of the network. The Host can stop the network from sending data by issuing a "Network Out of Service" command. Administrative frames will still be routed.

Summary - Bringing Up a New Network

The steps to bring up a new network are:

1) - Nodes send Join Network Requests to learn about their position in the network.

2) - The nodes register their knowledge of the network with the Host if possible.

3) - Route tables are generated and are disseminated to each node by the Host.

4) - The network is placed in service.

Once the network is operational normal communication is enabled. For two units to communicate the following steps must be performed:

1) A transceiver must have a route to the node it wishes to communicate with. For the host this could be any node, for a remote this must be the host.

2) The data must be formatted for transmission.

3) A link must be started with the routing/end-point node.

4) The data is physically transferred.

Route Determination

A transceiver with data to transmit checks its vector table for a path to the destination node. If no path is available, or a node with that ID does not exist, then

the vector table will contain a 255 for the routing node ID (primary and alternate). See, e.g. the Vector tables of Figs. 16-23 for nodes 1-8 of the network of Fig. 15. This "255" ID is an error condition and the node must notify the host. The node puts together a 'No Route Available' message that it sends to the host (if possible). Included in the message is the header from the undeliverable data (if the node is in debug mode the entire data message is included). If the data arrived over the RF link and this node is the final destination, then the vector table will contain a 254 for the routing node ID. This is a failsafe, since the destination address should previously have been examined and a determination made that this is the final destination.

If this is the first attempt to transmit the message the primary routing node ID and the hops will be read from the table and placed in the frame header as the intermediate ID and hop counter. If all of the attempts to transmit over the primary route have been exhausted then the alternate route is used.

Data Formatting

Once a route has been determined, the data must be formatted for transmission. The data is broken into frames with a header and trailer. The header contains the information show in Fig. 24.

The trailer is a 16bit CRC of the data.

The data is broken into frames and numbered in sequential order starting with zero. It is important that the sequence number not be reset for each data stream from a source. For example, if a remote device sends a small number of packets on a frequent periodic basis, the sequence number should not be reset after each data

stream. This could lead to packets from one data stream being mixed with packets from another stream if the first stream is delayed in arrival. For example, if a message requires 10Kbytes of data to be transmitted and a transceiver can only buffer 8Kbytes, then the message will need to be broken into parts. The first 8Kbytes might contain sequence numbers 0 through 120, the second half of the message, which can commence after the data from the first half is forwarded, will begin with sequence number 121. The individual nodes routing the message are not interested in the progression of sequence numbers from one part of a session to the next, they are useful to the endpoint trying to reconstruct a message.

The message type is set in accordance with the payload of the packet.

Two priorities of message exist, normal and high. For a data packet to have a high priority the device the transceiver is attached to must be intelligent so that it can set the priority or the transceiver must be configured to send all of its data at high priority. A device such as a smoke detector may have its transceiver set to send all of its data at high priority. When the data arrives at a node, it is placed in the appropriate queue for retransmission. When the transceiver is free to transmit, it will transmit messages in the priority queue before those in the normal queue. If a transceiver has a high priority message to transmit and the transceiver it needs to communicate with is linked to another transceiver, the message will have to wait. There is no facility to interrupt a link. Due to existing links, a high priority message my not actually get delivered any sooner than would a normal priority message.

The effect of having a high priority message to send depends on how the message arrives at the node. Referring to Fig. 25, If the high priority message is

generated either by the device attached to a node currently in communication or by a diagnostic on that transceiver then the transceiver will stop the current transmission by setting the data position bits to the last frame and transmit that frame as the last whether it was or not. Following that frame it will transmit the high priority frame(s) and then start with the data in its buffer as if it were the beginning of a message. The ack type refers to the local acknowledgement of the two transceivers in communications. End-to-end acknowledgement is only available when the peripheral that sent the data is capable of handling the acknowledgement, the transceivers do not process it.

The number of hops from the originating node to the destination is part of the vector table. Since there are primary and alternate routes and the number of hops can be different for the different routes the hop count is seeded with the larger of the values (i.e., a worst case route is assumed). This number is used to insure that a packet does not get caught in a loop. As shown in Fig. 26, each hop the hop counter is decremented; it is not decremented for retransmissions. If the hop count becomes zero before reaching the destination the packet will be discarded and a 'Time to live expired - Packet'Discarded' message will be sent to the host.

<u>End Point Data Interface</u>

It is instructive to examine how data enters and exits the ZLRT9600 network even before we examine how it gets across the network.

<u>Remote Serial Port Interface</u>

The peripheral attached to a remote ZLRT9600 passes data to the transceiver via the serial port. The serial data has no formatting. The data

formatting described in section 4.2 takes place before transmitting over the RF link. Conversely, the formatting is removed before the data is sent to the peripheral.

Host Interface

The host must be able to process data to and from multiple remotes, this means that the host must be able to discern the source of data streams, and specify recipients. The previously described header is used for differentiating data. When the transceiver has data to send to the host PC, the transceiver executes a number of steps, including:

1) sending the data to the local serial port, this causes an interrupt to the host PC and it begins reading the data

2) the data arrives at the host fully framed with the header and trailer

3) the host PC examines the message type to determine if the frame contains administrative or data information

4) the host PC does not need to qualify the data integrity since that has been done already by the transceiver. The host PC examines the source ID and the data position to determine where the data came from and its place in the over all message.

When the host has something for the transceiver it creates a header in which three fields have been filled in:

a) the destination ID

b) a type field which identifies following message as either administrative or data

c) the size of the message payload.

The host then sends the header and message payload to the transceiver.

Physical Interface

It is important to note that the physical interface is far simpler than in the previously described Zeus point to point and host to multipoint systems. Since the header identifies the message type there is no need for a command mode triggered by DTR. In addition, datagrams are autonomous units, which makes it unnecessary to use DCD for session indication.

Session Initiation – Single-part

Once the route has been determined and there is data to transfer the transceiver initiates a session with the next node in the route. This node's ID has been placed in the frame header as the intermediate ID and the final destination's ID is in the destination field. As shown in Fig. 27, if a link cannot be established with in a predetermined number of times, the intermediate ID is changed to the one for the alternate route and the linking process begins again.

If an attempt to transmit on both routes is unsuccessful and there is room in the transceiver's buffer, the transceiver will go into receive mode and accept packets from other nodes. At the end of this receive period the transceiver will transition back to attempting to transmit first with the primary and then with the alternate. A configurable parameter determines how many times an attempt is made to transfer data. Once the number of attempts has been reached the maximum configured, a maximum delivery attempts exceeded error message is constructed with the header from the undelivered message included and sent to the host. The data is then deleted. If the maximum delivery attempts is set to 255 the maximum attempts are

infinite. If the direction that is blocked is to the host, the error message will go through the same attempts aging procedure.

Session Initiation – Multi-part

When a session is initiated as described above the data will be transmitted until either there is no more data to send or the receiver's buffer fills. If the receiver's buffer fills it will need to stop receiving, transition to transmit mode and clear its buffer before it can accept more data. The transceiver that initiated the session will be told to stop sending data. This is a multi-part session.

The details of a multi-part session are as follows:

- transceiver A initiates a session with transceiver B.

- transceiver A begins transmitting data to transceiver B, and keeps a count of the number of packets transferred

- transceiver B's buffer fills to a high watermark which is at least one packet less than a full buffer

- transceiver B sends a 'Cease Transmit' command to transceiver A. This command is functionally equivalent to the software flow control command XOFF.

- transceiver A goes into a 'waiting to send mode'

- transceiver B initiates a link with the next node in the route to the destination, transceiver C.

- transmitter B sends the contents of its buffer until either it is empty or told to 'Cease Transmit'

- to allow for packet forwarding, transceiver A will wait the packet transmit time for each of the packets it transmitted to transceiver B and then attempt to

initiate a link with transceiver B. If it cannot link, then it will wait a random set-back and try again. If it cannot link with in a configurable number of attempts a maximum delivery attempts exceeded error message is constructed with the header from the undelivered message included and sent to the host. The data is then deleted. (NOTE: No attempt is made to try the alternate route.)

- if transceiver B was able to transfer enough data to go below its buffer low watermark then it will link to transceiver A

- transceiver A completes sending its data and in the last packet sent sets the data position bits to the last frame

- transceiver B will attempt to initiate a link with transceiver C

- if transceiver C accepts the link (It has room in its buffers) the data will transfer. There is no linkage between this transmission and the last between these two transceivers.

<u>Session Initiation – Multi-path (Future Implementation)</u>

Multi-pathing attempts to overcome the wait inherent in multi-part sessions. Since there is usually a primary and secondary path available to each transceiver and waiting for data to propagate is very detrimental to throughput a method of using both paths for large transfers has been developed. In a multi-path transfer the data is transferred as it is for a multi-part transfer, but when the transmitting transceiver is flow controlled a second session is initiated through the alternate path. See Fig. 28. When the alternate path is full the transmitting transceiver can switch back to the primary route. Note that if the primary path becomes temporarily blocked packets may not be received in the right order. It is easy to see where the throughput of

transceiver A is increased by multi-path transmission. If the size of the data being transferred to nodes B and C is the same then the transfer times should be the same. If B can acquire a link in the same amount of time that A acquires its link to C then both A's transfer to C and B's transfer should complete at the same time. Transceiver A can now initiate a link to transceiver B again if necessary. A simple example of this can be seen in the Fig. 29.

Multi-Source Transfers

When transferring small amounts of data the receiver's buffer will not fill as in the multi-part and multi-path examples above. When a transceiver with room in its buffer is unable to link to the next layer nodes it waits a standoff time and tries to send again. During this standoff another transceiver may contact this transceiver and send it additional data for forwarding. Since the headers uniquely identify the data there is no ambiguity in accepting data from multiple sources. When a link is made to the next layer, all of the data is transferred.

The bi-directionality of data transfers presents an interesting application. In the following example illustrated in Fig. 30:

1) transceiver A sends data to transceiver B.

2) Transceiver B attempts to bring up a link to transceiver E and is unsuccessful.

3) Transceiver B goes into the standoff wait. Transceiver C links to transceiver B and transfers data.

4) Transceiver B attempts to bring up a link to first transceiver E and is unsuccessful.

5) Transceiver B goes into the standoff wait. Transceiver D links to transceiver B and transfers data.

6) Transceiver E links to transceiver B and transfers data. Transceiver B accepts the data from E

7) Transceiver B transfers the data it has for that layer to E.

8) Transceiver B attempts to bring up a link with transceiver A to transfer the data newly received from E.

In this example, it is important that transceiver B examine the destination of each packet of data to avoid transferring back to transceiver E the data just received. Data cannot be blindly transferred as in the point-to-point and host-to-multipoint networks.

If transceiver F had linked instead of E and F was the alternate then B would not have transferred A, C, and D's data to F. This is because even though F is in the correct direction it may be significantly slower than taking the primary route.

Scenarios

This section provides several scenarios that demonstrate the functionality of the phase one implementation of Store and Forward. See Fig. 31.

Scenario 1 – A to B

In this scenario transceiver A sends a single frame message to transceiver B to be forwarded. Assume transceiver A is idle until it receives data from its attached dumb device.

1) A receive interrupt occurs on the local serial port of transceiver A.

2) The local serial port receive function places the raw data on the local port in queue.

3) The executive decides that data can be transmitted based on the receive/transmit cycle timing and calls the framing system.

4) The framing system puts on the header and calls the routing system to get the intermediate node ID. The header gets the sequence number, position bits, intermediate node ID, etc. and fills them in.

5) The queuing system places the data on the RF out queue for transmission.

6) The RF transmit system calls the tuning system and proceeds through the linking phase.

7) Once an ACK has been received to the linking frame the RF transmit system tunes for transmit.

8) The RF transmit system sends the data.

9) Transceiver B receives the data. Transceiver B examines its RF out queue for data to go to transceiver A.

10) Transceiver B finding no data for transceiver A creates an ACK frame and places it on the RF out queue.

11) Transceiver B sends the ACK to transceiver A.

12) Transceiver A receives the frame and verifies it.

13) Transceiver A notes the ACK and flushes the frame it transmitted to transceiver B.

14) Transceiver A returns to idle mode.

Variation 1 – Transceiver A does not receive the ACK

1) Transceiver A does not receive the ACK with in the specified bad data hops and goes back into linking mode.

2) When transceiver A links back with transceiver B it retransmits the frame. This will continue until transceiver A receives the ACK or it gives up after a configurable number of retries.

3) Transceiver B sends each retransmission on never checking for duplication.

4) If transceiver A reaches the maximum retransmissions it will flush the RF out queue of the frames remaining for transceiver B. If the maximum retransmissions are set to infinite transceiver A will try to transmit forever.

Variation 2 – Transceivers A and B link, but B never receives data.

1) Transceiver A sends data to transceiver B who never receives it. Transceiver B will wait the specified bad data hops and goes back to idle mode.

2) Transceiver A will also wait the specified bad data hops and go back into linking mode.

Scenario 2 – A to B to A

In this scenario, transceiver A sends a single frame message to transceiver B to be forwarded, transceiver B has a multi-frame message to send to transceiver A. Assume transceiver A is idle until it receives data from its attached dumb device.

1) A receive interrupt occurs on the local serial port of transceiver A.

2) The local serial port receive function places the raw data on the local port in queue.

3) The executive decides that data can be transmitted based on the receive/transmit cycle timing and calls the framing system.

4) The framing system puts on the header and calls the routing system to get the intermediate node ID. The header gets the sequence number, position bits, intermediate node ID, etc. and fills them in.

5) The queuing system places the data on the RF out queue for transmission.

6) The RF transmit system calls the tuning system and proceeds through the linking phase.

7) Once an ACK has been received to the linking frame the RF transmit system tunes for transmit.

8) The RF transmit system sends the data.

9) Transceiver B receives the data. Transceiver B examines its RF out queue for data to go to transceiver A.

10) Transceiver B finding data for transceiver A removes a data frame from the RF out queue and sets the ACK in the header.

11) Transceiver B sends the data to transceiver A.

12) Transceiver A receives the frame and verifies it.

13) Transceiver A notes the ACK and flushes the frame it transmitted to transceiver B.

14) Transceiver A decodes the rest of the received frame.

15) The frame is destined for transceiver A; transceiver A calls the framing system to remove the framing and buffering system to place the raw data on the local port out queue.

16) The serial port driver transfers the data out the local serial port.

17) Transceiver A examines its RF out queue for data to go to transceiver B.

18) Transceiver A finds no data and calls the framing system to create an ACK frame. The buffer system places the frame on the RF out queue.

19) Transceiver A sends the frame to transceiver B and goes into receive mode.

Scenario 3 – PC to A to B

In this scenario, transceiver A is connected to the host PC. The host creates a message destined for transceiver C. Transceiver A transfers several frames of data to transceiver B before an RF blockage keeps the remaining frames from transferring. Transceiver A notifies the PC that it is unable to transmit the data. The PC decides that enough time has elapsed with out being able to transmit the data and orders transceiver A to flush the data remaining for B.

1) A receive interrupt occurs on the local serial port of transceiver A.

2) The local serial port receive function places the data on the local port in queue.

3) Since transceiver A is attached to an intelligent device, it decodes the frame header to determine if the frame is administrative or data. Upon determining

that this is a data frame if calls the framing system to divide the data into transmission frames.

4) Transceiver A calls the routing system to fill in the header routing information, sequence number, position bits, intermediate node ID, etc...

5) The buffer system is called to place the buffer on the RF out queue.

6) The RF transmit system calls the tuning system and proceeds through the linking phase.

7) Once an ACK has been received to the linking frame the RF transmit system tunes for transmit.

8) The RF transmit system sends the data.

9) Transceiver B receives the data. Transceiver B examines its RF out queue for data to go to transceiver A.

10) Transceiver B finding no data for transceiver A creates an ACK frame and places it on the RF out queue.

11) Transceiver B sends the ACK to transceiver A.

12) Transceiver A receives the frame and verifies it.

13) Transceiver A notes the ACK and flushes the frame it transmitted to transceiver B.

14) Transceiver A examines its RF out queue for data to go to transceiver B.

15) Transceiver A finds data on the RF out queue for transceiver B.

16) Transceiver A's RF transmit system sends the data.

17) Transceiver A does not receive the ACK with in the specified bad data hops and goes back into linking mode.

18) Transceiver A is unable to link back with transceiver B. This continue until transceiver A's attempts to link reaches the maximum number of retries.

19) Transceiver A calls its administrative system to generate an administrative frame, maximum delivery attempts exceeded, for the host PC. Transceiver A is not configured to automatically discard the remaining frames for transceiver B.

20) The administrative system calls the framing system to add a header and CRC16 to the frame.

21) Next, the buffer system is called to place the frame on the local port out queue.

22) The local port driver sends the frame out the local port.

23) The host PC receives the frame and decides to abort the attempts at linking to transceiver B.

24) The PC creates an administrative frame to inform transceiver A to delete the remaining frames for transceiver B.

25) Transceiver A's local port driver places the data on the local port in queue.

26) Since transceiver A is attached to an intelligent device, it decodes the frame header to determine if the frame is administrative or data. The frame is administrative and is passed to the administrative system.

27) The administrative system decodes the message and calls the buffer management system to delete the frames for transceiver B.

<u>Summary</u>

Messages can be transferred from an attached device to the transceiver and on out over the RF link.

Messages that arrive over the RF link can be destined for the attached device.

Messages that arrive over the RF link may be intended for forwarding to another transceiver.

<u>Data from attached device</u>

1) Data is received from the attached device.

2) The route to the destination is determined.

3) The data is formatted into frames for transmission and placed on the RF transmit queue.

4) A frame is taken off the head of the transmit queue and a link is established with the destination.

5) Frames are taken out of the transmit queue and transferred until

a) there are no more frames for this destination

b) the link goes down,

c) or the unit is flow controlled.

<u>Data to attached device</u>

1) Frames are received over the RF link, validated, and the destination examined.

2) Frames for this destination is striped of the header and trailer and placed in the local RS232 output queue.

Forwarded data

1) Frames received over the RF link are validated and the destination examined.

2) Frames to be forwarded will have their header updated with the ID of the next intermediate node and the hop count will be decremented and checked for expiration.

3) The data is placed on the RF transmit queue.

Host Interface

The host communicates with the transceiver as an intelligent device by sending framed messages rather than raw data.

The host creates a header for the command or data payload to be sent. The type and size of the payload are filled in.

The host sends the message.

Error Detection and Recovery

Errors can occur and be detected at various points in the network. Since the majority of the devices connected to transceivers are not intelligent they cannot be relied upon to assist in either the detection or correction of errors. The burden of error detection and correction falls on the transceivers and host computer. The limited processing power and buffer space available in the transceivers limits the amount of error recovery that they can perform. In general the transceivers will insure the integrity of the data as it passes from layer to layer, the previous layer

transceiver cannot recover from a failure in the next layer transceiver if the data transfer has already been acknowledged.

For example, transceiver B fails or the link to it is blocked after acknowledging packet 5 of 10 from transceiver A. Transceiver A attempts to link to the alternate route to transfer the remaining packets but does not retransmit packets 1 – 5. If transceiver B failed then the host will receive only packets 6 – 10, if transceiver B's link to A was temporarily blocked then the host will receive all ten of the packets.

If, in the previous example, transceiver A assumed that the data was lost since it had not transferred all of it and it was unable to communicate further with B, it could retransmit it on the alternate route. In this scenario, transceiver A needs to save all of the data until the entirety of it is acknowledged. The data could still be lost if transceiver B fails after A has completed sending its data.

Data Errors

When a data error is detected, two scenarios are possible; either the problem can be corrected with the data available in the forward error correcting logic or it cannot. When the problem cannot be corrected, retransmission of the frame must be requested by negatively acknowledging (NACKing) the frame as shown in Fig. 32. If the frame can not be successfully retransmitted after a configurable number of tries then the transmitting node will create an excessive retransmission error message and route it to the host with the data header. The transmitting node will then delete the data. (NOTE: the number of retransmissions could be set to infinite (255), in which case attempts will be made forever.)

In the Fig. 33, for an Undeliverable Data Error,

1) nodes A and B link and successfully transfer a frame of data C.

2) the second frame of data in the stream C can not be delivered with in the maximum configured attempts.

3) node A creates a diagnostic message – excessive retransmissions, which contains the header from the undeliverable frame.

4) node A deletes the remaining C data. Node A does not attempt to find an alternate route since it cannot be sure what happened to the frames which were successfully transferred to B. If node A did transfer over an alternate route the final destination might not be able to reconstruct the message.

5) node A links to B and attempts to transfer message D.

6) after the maximum attempts have been made to transfer the data with out success, node A drops the link to B

7) node A links with the alternate node for this data and transfers the message D. Node A went to the alternate route because none of the data had been successfully transferred to another node.

Frame Sequence Errors

Duplicate Frames

Intermediate nodes, those routing data, handle frame sequence errors differently then the final destination. At an intermediate node if a duplicate frame is received it is not detected. The intermediate node will acknowledge the frame again and send it on. At the final destination the duplicate frames will be detected and discarded.

Duplicate frames can occur when the acknowledgement to a frame is not received. If an acknowledgement for the original frame is not received with in a given time the original frame will be retransmitted causing a duplicate frame.

Missing Frames

Missing frames can occur in three ways:

1) A portion of the data is transferred and then the originator is flow controlled. One of the intermediates along the route receives and acknowledges the data and then crashes losing all of the data. The originating node has flow control lifted and the remainder is passed to the destination. The destination will detect that frames are lost because the position bits will show no beginning of message. See Fig. 34.

The configuration of the node will determine if the incomplete message is deleted or passed on to the attached device. In either case a diagnostic message – missing frames - will be generated and sent to the host.

2) Data is being transferred from node A to B to C. Node B flow controls node A, forwards all of the frames to node C, and crashes. Node A is unable to reestablish a link to B, flushes the remaining frames and sends a 'could not link with a node after N attempts' error message to the host. The frames that had been transferred by node B prior to crashing are passed to the attached device by the destination node. If the remaining frames are not received with in a configured amount of time the destination node will create a 'missing frames' message and send it to the host. If the remainder of the frames arrive after the 'missing frames'

message is sent the frames will be treated as described in scenario one above. See Fig. 35.

Missing intermediate frames from a message is possible with the phase one implementation but is more likely with later implementations that will support multi-path transfers. For intermediate frames to be missing the transmission of these frames must be interrupted in such a way that they are deleted with out causing the remaining frames to be flushed. If a link were to go down long enough for a node to decide that it was not coming back and cause it to flush the remaining frames, this condition could propagate back through the network. This situation points out the importance of careful network configuration. See Fig. 36.

If missing frames are detected and the final destination is not the host, then an error message – "missing frames" - is sent to the host with the information from the last packet header. The destination transceiver can be configured to either discard a sequence of packets if some are missing or send them to the connected device.

From the vantage point of a remote end point, it is easy to tell when data is missing. The first frame in a message will have the data position bits in the header set to that of the first frame. All intermediate frames will have the bits set to intermediate. The last frame will have the data position bits set to the last frame. The variable current_sequence is used by the remote end point to track the arrival order of frames. Initially this variable is set to the don't care value 255. When a first frame is received the remote sets the current_sequence equal to the sequence number in that frame, in the example above this would be 100. As each frame is

received its sequence number is compared to the current_sequence +1. If they do not match then an error has occurred. When the last frame is received the current_sequence is set back to the don't care value. This simple system works well since all of the data destined for a remote was originated from the host.

The host can accept data from multiple nodes and data streams from multiple sources may be intermingled, because of this the host is required to keep a current_sequence number for every available node. The behavior of this variable is the same as for the remote. For example, if the host receives data as shown in the following table,

Source ID	Sequence Number	Data Position
5	101	beginning
5	102	intermediate
5	103	intermediate
5	104	final
23	17	beginning
23	18	intermediate
23	20	intermediate
23	21	intermediate
41	78	beginning
41	79	final
23	22	intermediate
23	23	intermediate
23	24	final
37	92	begin & final
6	1	intermediate
6	2	intermediate
6	3	final

the current_sequence number for each source node, if the host transceiver is programmed to send on incomplete data, would be:

Source ID	Current_sequence								
5	255	101	102	103	104	255			
23	255	17	18	20	21	22	23	24	255
41	255	78	79	255					
37	255	92	255						
6	255	1	2	3	255				

If the host transceiver was programmed to discard data once missing data was detected then the lines for source ID 23 and 6 would be:

Source ID	Current_sequence								
23	255	17	18	255					
6	255								

Data Structures

The data structures involved in communications are:

1) local port serial data buffer
2) input
3) output
4) RF data buffer
5) input
6) priority out
7) normal out

The data structures are all queues (first in first out). The local port serial buffers (with the exception of the host transceiver) contain raw data, i.e., straight data no header or trailer. There is no need to segregate the data in these queues, it simply passes one direction or the other. The RF buffers all contain formatted data (header and trailer). In the RF buffers it is important to know the boundaries of a frame for routing purposes. Since the data is formatted in these buffers it is easy to find the

frame boundaries. The queue head pointer marks the beginning of the first frame. The format of a header is known and contains the length of the attached data. With this information the end of the frame and beginning of the next frame can easily be determined.

Local Port Serial Data Buffer

Input To The Transceiver From A Remote Peripheral

Data coming in the serial port to the transceiver is placed in the local port receive buffer. The receive buffer is a circular FIFO. If flow control is enabled, when the amount of data in the buffer reaches the high watermark the transceiver will flow control the peripheral. Once enough data has been transmitted from the buffer for the amount to go below the low watermark the transceiver will allow reception again. The use of two separate watermarks as shown in Fig. 37 protects against needing to flow control on the first data received immediately after releasing flow control.

Output To The Remote Peripheral

Data arriving from the RF link for this transceiver is placed in the serial output buffer. If the remote peripheral flow controls input then the transceiver will accept data into the serial output buffer until it is full. Once the buffer fills, the RF section of the transceiver is told to flow control incoming data. Once space is available in the output buffer the RF section is told to release flow control.

RF Data Buffers

Input To The Transceiver

Messages that arrive over the RF link are examined to see if they should be processed or ignored. If the vendor ID and network ID are correct then the

destination ID is examined to see if this node is the end point. If this node is the end point then the data is fully validated. The message type is examined and data is placed in the serial output buffer; administrative messages are processed.

If the destination ID is not this node's the intermediate ID is examined to see if it is this node's. If it does not match then the message was not meant for the node and it is discarded. If the ID does match then the message is fully validated and placed in either the normal or priority RF output buffer based on the header priority field.

Output Buffers

Two RF output buffers exist, a high priority buffer and a normal priority buffer.

Messages in the high priority buffer are sent preferentially to those in the normal buffer. When the transceiver transitions to transmit, the priority buffer is examined and any messages in it are sent. If there are no messages in the priority buffer then what ever is in the normal buffer is sent.

Messages for output over the RF link can originate from one of three places:

1) the peripheral attached to the serial port
2) the RF link
3) a locally generated administrative message.

All messages originating from the peripheral attached to the serial port have the same preconfigured priority. A device such as a smoke detector would have a high priority for all of its communications; a less important device would have the normal priority. Once the data has arrived it will be framed with a header and trailer, forward error corrected, and placed in the RF output buffer.

Referring now to Fig. 38, messages which arrived over the RF link and are to be forwarded must have their forwarding information updated. The destination ID is read and used as an index into the vector table to find the next intermediate. The intermediate value in the header is updated with this node ID, even if that causes it to be the same as the destination. The hop counter is decremented and checked to see if the time to live has expired. If it has expired, then an error message is sent to the host with the header from this frame. The frame is then deleted. If the time to live has not expired then the CRCs are recalculated and the frame is placed in the appropriate RF output buffer.

If the message is an internally generated administrative frame, it is processed identically to data coming in the local serial port. A header and trailer are created, and the message is placed in the appropriate output buffer.

Summary

Five buffers are used for moving data between the RF and local ports. The buffers for RF are:

- RF in
- RF priority out, and
- RF normal priority out

The buffers for the local serial port are:

- Serial in and
- Serial out

The structure of the five buffers are identical queues. Each buffer has a high watermark and a low watermark to aid in flow control. The messages on the RF

queues are always framed, messages on the local port queues are framed for communications with the host and unframed for communications with an ordinary peripheral.

Administration System

The administrative system provides a command interface that is accessible both through the local serial port and over the network. Messages to and from the administrative system are framed. Referring to Fig. 39, the frame header identifies the contents as being administrative data. With in the frame's payload are one or more administrative messages. Only one type of administrative message may exist in the frame. The administrative message types are:

1) Register request
2) Register response
3) Diagnostic request
4) Diagnostic response
5) Discovery request
6) Discovery response
7) Network management request
8) Network management response
9) Unsolicited diagnostic
10) Error report
11) Host command

These messages each have their own header that identifies the specific administrative message and the length of the message where applicable. Messages

that have a predefined length, such as register messages, do not need a length field.

Register Request/Response

The register system is designed to provide the user with the maximum flexibility to configure and monitor the functionality of the transceiver. The system also provides for three levels of password protected access to the configuration data. These three levels represent user, OEM, and factory access. The factory access allows Zeus Wireless personnel to configure every aspect of the radio. The OEM access is a subset of the factory access which allows the OEM to provide a level of customization. The user access allows the end user to perform a limited amount of customization specific to their installation.

The password authorization is performed on two levels. On the host PC a password administration system exists which allows the system administrator to create users and assign them passwords and levels of access. A second password system exists on the transceiver itself. The transceiver has passwords for the user, OEM, and factory that are separate from those administered by the host PC. The user and OEM passwords are configurable through the register system; the factory password is not.

The administrative frame to request for the contents of a group of registers is composed of: the administrative message type - register request, first register number, second register number, etc... The response would be: register response, first register number, first register contents, second register number, second register contents, etc...

The new registers for store and forward are:

	The number of times to try a route
	The number of times to cycle between trying the primary and alternate routes
	Whether or not to pass messages with missing frames
	Join network retry timer
	Network operational flag – read only

Diagnostic Request/Response

Discovery Request/Response

Network Management Request/Response

Like a register request the host can send a request. A request can cause the transceiver receiving it to perform a function and report the results or to simply return a status. The administrative frame for a request is composed of: the administrative message type - xxxx request, first request number, second request number, etc... The response would be: xxxx response, first response number, size of response, first response contents, second response number, size of response, second response contents, etc...

Unsolicited Diagnostic And Error Report

Unsolicited diagnostics and error reports are both sent to the host in response to the occurrence of an event. A diagnostic differs from an error in that what it reports is informational. The host uses the information in the diagnostic to monitor the health of the network and make decisions about the management of the network. An unsolicited diagnostic is sent when a threshold for a type of event has been reached.

Error reports are generated when an error condition has been detected. Depending on the configuration of the transceiver reporting the problem, the transceiver may initiate an action when an error is detected or simply report it to the host allowing the host to initiate any action necessary.

Host Command

A host command can be one of the following message types; discovery or network management. A command is simply a message that does not require a reply. The format of a command is identical to that of a request.

Network Administration

Each aspect of the network management system has two components, one, which requires a network administrator to manage, and one that is automatically controlled through a task. The components of the network management system are: network configuration management,

1) determines the configuration of the network, and
2) monitors the addition and deletion of network nodes.
3) network performance,
4) fault management, and
5) network security.

In general, the simpler the network topology, the simpler the network management. The network can be administered either through the ZNet utility supplied by Zeus Wireless or a custom network administration can be developed using the API provided.

Network Configuration Management

The configuration of a network depends on:

1) The physical topology to which the network must adhere. For example, is the network all on one floor of a building, multiple floors, multiple buildings?

2) The medium used for network transmission. In the case of the ZLRT9600 transceiver (e.g., transceiver 10) this is line of sight RF.

3) Environmental effects such as radio interference from other emitters or physical blockage.

4) The type of network desired, i.e., point-to-point, host to multipoint, or host to multipoint with store and forward routing (repeater).

5) The ability to coexist with neighboring networks.

6) The functional and logical addressing of the network nodes. For example, it might be beneficial to have logical subnetworks to aggregate devices for easier management.

Since the physical plant aspects of a network, items 1 through 3 above, must be examined in the context of a specific deployment they will not be covered here. The most efficient type of network is typically the most direct. The point-to-point and host to multipoint topologies are more efficient than a store and forward network that requires intermediate nodes. The more intermediate nodes there are the more delay there will be in transmission and the more chances for failure. The more alternate routes however, the more fault tolerant the network.

The ability of one ZLRT9600 network to coexist with a neighboring ZLRT9600 network can be assured by two things:

1) choosing frequency hop tables to avoid collisions, and

2) each network having its own unique network ID.

A network can be logically divided into a series of subnetworks by Znet to provide either a functional or physical dichotomy. It may be helpful to organize all devices of a similar type on a single subnetwork or all devices on a specific floor of a building might be grouped together.

Network Performance

The performance of a network can be optimized for the most common type of network traffic and network topology. An example of optimizing for a typical traffic pattern is setting the fall-back period to wait after a collision before attempting a retransmission differently for short frequent burst traffic and long infrequent burst traffic.

Fault Management

A fault in the network can be caused in two ways, either by a node in the network failing or by a physical blockage of the RF. If the network is a point-to-point topology the only remedy available is to physically fix the problem. In a store and forward network alternate routes frequently exist and can be used to work around and report faults. A host to multipoint network can be configured to have alternate routes at which time it becomes a store and forward network. For example, the network of Fig. 40 has store and forward interconnections between all of its nodes; this allows each node to
have a primary and secondary route to the host.

The fault management task receives diagnostic reports and can use the information they provide to change vector tables to avoid node outages. Diagnostic messages are either sent unsolicited from the transceivers or requested by the fault management system.

Unsolicited Diagnostics

When a node detects an error it will create an error message whose destination is the host. The node will do this even if it knows that there is no current path to the host. When a path becomes available the message will be transmitted. Messages of this type are:

Frame Errors	
Missing frames	Not all of the frame sequences were received.
RF Transmission Errors	
Excessive retransmissions (bad hops) requested by this node requested of this node	The maximum frame retransmission was reached, i.e., the frame was NACKed the maximum number of times. This message can be reported either by the one doing the NACKing, the one being NACKed, or both.
Could not link with a node after N attempts	The transceiver reached the maximum number of attempts to try and link.
Low RSSI	The partner transceiver could not adjust its RSSI up to an acceptable level.
High RSSI	The partner transceiver could not adjust its RSSI down to an acceptable level.

Excessive no data hops	
Messages received from different network	A message was received with a network ID that does not match this network. This is an indication that a neighboring network has some hop frequencies that overlap. If a large number of frequencies are detected as overlapping it would be prudent to adjust the hop table for one of the networks.
Serial Port Errors	
Flow control does not stop data – buffer overflow	Flow control has been asserted (either hardware or software) to the attached device and the device has not stopped sending.
Excessive parity errors	The UART has detected excessive parity errors.
Excessive framing errors	The UART has detected excessive framing errors.
EEPROM Errors	
User data	The CRC 16 for the user data is incorrect.
OEM data	The CRC 16 for the OEM data is incorrect.
Zeus data	The CRC 16 for the Zeus data is incorrect.
Routing Errors	

No route available	Error message sent to host to report a route to the requested destination was not available.
Request to route to non-existent node	A frame arrived whose destination when indexed in to the vector table yielded a 255 for the intermediate node ID.
Time to live expired - discarded packet	The hop count associated with the frame reached zero before the frame reached its final destination.
Maximum delivery attempts exceeded	This message indicates that the maximum attempts have been made to deliver a message.
Excessive retransmission	The maximum frame retransmission was reached, i.e., the frame was nacked the maximum number of times.
Missing frames	Not all of the frame sequences were received.

Solicited Diagnostics

Two sets of diagnostics exist, network level and ZLRT9600 component level. The host processor performs network diagnostics by keeping a table of node statuses. On a configurable periodic basis the host will request status. This status message serves the following purposes,

- it lets the host know that the node is alive and able to communicate,

- what its operational health is, and

- the status of its communications with neighboring nodes since the last report.

If a device does not respond to the poll or it declared itself bad in the node status message then the host will mark the node bad.

The host has several options available for reporting and handling failures:

- It can flash an alert on the host console.

- It can page the network operator.

- It can send e-mail to the network operator

- It can, if alternate routes exist, update the vector tables to try to route around the problem.

The ZLRT9600 is a field replaceable unit and as such any diagnostic will result in a simple pass/fail indication. If the unit fails it must be removed and replaced.

Network Security

Network security is ensured by three things,

rapid and random frequency changes (i.e., "hopping") over a large set of frequencies,

unique vendor, network, and node IDs are transmitted in each frame of data, and

nodes must register with the host computer to become part of the network.

Another option to increase security is data encryption. The firmware of the ZLRT9600 provides a hook where data encryption can be inserted. Since this is an open solution any level of data encryption required can be inserted.

ZNet

ZNet is the Zeus supplied PC application for network management. The graphical view provided by ZNet allows the user to view a representation of the network. The user may assign different shapes and colors to network elements to categorize them. Moving the cursor above an element of the network provides a pop-up summary status box about that element which includes:

- the device type,
- the node ID,
- the node status, and
- the vector table

Right clicking on the network element brings up a list of functions that can be preformed on the element. The diagram of Fig. 41 shows different shapes representing different devices and logical grouping by floors.

The Zeus API Functions

ZEUS Wireless, Inc. provides a set of Application Programming Interface to interface with the ZLR9600 firmware. These Zeus API are used for creating a custom interface in the Win32 environment. These functions are written in Visual C++/MFC, compiled and build as a dynamic link library.

The ZEUS API consists of these general functional categories:

- Transceiver Data
- Configuration and setup
- Set and Get Hardware registers
- Initialization
- Load and store parameters

The following Zeus API's are defined:

wCi_Open

Call this API function to open the Communication port and connect to the port if it is available.

Syntax	int wCi_Open (LPCTSTR pszCommPort)
Parameters	*pszCommPort*: Points to the string name of the communication port (COM1, COM2, COM3 or COM4). The string must have a terminating null character.
Return Value	If the communication port is ready to transmit and receive data. 1: The communication port can not open. 2: Fail creating the Read – Write thread.
Example:	int status = wCi_Open ("COM1");

wCi_Close

Call this API function to close the Communication port, which is opened with the function wCi_Open.

Syntax	int wCi_Close (void)
Parameters	None
Return Value	Always 0
Example	int status = wCi_Close

wCi_CheckTxQue

Call this API function to check if the transmit FIFO has been emptied.

Syntax	int wCi_CheckTxQue (void)
Parameters	None
Return Value	The number of bytes contains in the transmit FIFO.
Example	int status = wCi_CheckTxQue

wCi_CheckRxQue

Call this API function to check if the receive FIFO has been emptied.

Syntax	Int wCi_CheckRxQue (void)
Parameters	None
Return Value	The number of bytes contains in the receive FIFO.
Example	Int status = wCi_CheckRxQue

wCi_ReadData

Call this API function to get data from the receive FIFO.

Syntax	int wCi_ReadData (unsigned char* *Buffer*)
Parameters	*Buffer*:Points to the buffer that receives the data.
Return Value	Returns the number of byte actually read from the receive FIFO.

wCi_WriteData

Call this API function to write the data to the transmit FIFO.

Syntax	int wCi_WriteData (unsigned char* *Buffer*, int *nBytes*)
Parameters	*Buffer*:Points to the buffer that transmits the data.
Return Value	Returns the number of byte actually read from the receive FIFO.

wCi_ReadDataEx

Call this API function to get data packet from the receive FIFO.

Syntax void wCi_ReadDataEx (CMessage * *Msg)*

Parameters *Msg:* Points to the CMessage object that receives the data.

```
class CMessage : public CString
{
public:
        UINT    m_nID;
        BYTE    m_byStatus;
        CString m_strText;
};
```

Return Value None

wCi_WriteDataEx

Call this API function to write the data to the Tx FIFO include the destination ID.

Syntax	int wCi_WriteDataEx (unsigned char* Buffer, int nBytes, unsigned int Destination ID)
Parameters	*Buffer* - Points to the buffer that transmits the data. *nBytes* - Number of bytes to write to the transmit FIFO. *DestinationID* - The destination Unit ID where the data will be transmit to
Return Value	Returns the number of byte actually write to the transmit FIFO.

wCi_SetDtr

Call this API function to modify the Communication control signal DTR.

Syntax	int wCi_SetDtr (int *DTR_Lead*)
Parameters	*DTR_Lead*
	0: Set the DTR control signal OFF.
	1: Set the DTR control signal ON.
Return Value	

wCi_SetRts

Call this API function to modify the Communication control signal RTS.

Syntax	int wCi_SetRts (int *RTS_Lead*)
Parameters	*RTS_Lead*
	0: Set the RTS control signal OFF.
	1: Set the RTS control signal ON.

wCi_GetDtr

Call this API function to get the status of the Communication control signal DTR.

Syntax	unsigned int wCi_GetDtr (void)	
Parameters	None	
Return Value	0:	The DTR control signal is OFF.
	1:	The DTR control signal is ON.

wCi_GetRts

Call this API function to get the status of the Communication control signal RTS.

Syntax	unsigned int wCi_GetRts (void)	
Parameters	None	
Return Value	0:	The RTS control signal is OFF.
	1:	The RTS control signal is ON

wCi_GetCts

Call this API function to get the status of the Communication control signal CTS.

Syntax	unsigned int wCi_GetCts (void)
Parameters	None

Return Value	0:	The CTS control signal is OFF.
	1:	The CTS control signal is ON.

wCi_GetDsr

Call this API function to get the status of the Communication control signal DSR.

Syntax	unsigned int wCi_GetDsr (void)	
Parameters	None	
Return Value	0:	The DSR control signal is OFF.
	1:	The DSR control signal is ON.

wCi_GetRi

Call this API function to get the status of the Communication control signal RI.

Syntax	unsigned int wCi_GetRi (void)	
Parameters	None	
Return Value	0:	The RI control signal is OFF.
	1:	The RI control signal is ON.

wCi_GetDcd

Call this API function to get the status of the Communication control signal DCD.

Syntax	unsigned int wCi_GetDcd (void)	
Parameters	None	
Return Value	0:	The DCD control signal is OFF.
	1:	The DCD control signal is ON.

wCi_BeginCommandMode

Call this API function to set ZRTL9600 firmware in command mode (start to receive command).

Syntax	void wCi_BeginCommandMode (void)
Parameters	None
Return Value	None

wCi_EndCommandMode

Call this API function to set ZRTL9600 firmware in data mode (start to transmit/receive data).

Syntax void wCi_EndCommandMode (void)
Parameters None
Return Value None

wCi_GetZeusPrivilege

Call this API function to set ZRTL9600 firmware in ZEUS privilege.

Syntax void wCi_GetZeusPrivilege (void)
Parameters None
Return Value None

wCi_GetOemPrivilege

Call this API function to set ZRTL9600 firmware in OEM privilege.

Syntax void wCi_GetOemPrivilege (void)
Parameters None
Return Value None

wCi_GetUserPrivilege

Call this API function to set ZRTL9600 firmware in USER privilege.

Syntax void wCi_GetUserPrivilege (void)
Parameters None
Return Value None

wCi_SmartModeEnable

Call this API function to modify transmit/receive data mode.

Syntax void wCi_SmartModeEnable (BOOL *sflag*)
Parameters *Sflag:* TRUE: Set transmit/receive data in session mode.
FALSE: Set transmit/receive data in normal mode.
Return Value None

wCi_GetCommEvent

Call this API function to get the communication port event.

Syntax DWORD wCi_GetCommEvent (void)
Parameters None
Return Value
The event mask of communication port:
EV_RXCHAR 0x0001 Any Character received.

EV_RING	0x0100	Ring signal detected.

wCi_RegisterCallBack

Call this API function to register the function module, which will be called if the communication port event occurs.

Syntax void wCi_RegisterCallBack (FARPROC CBfunction)
Parameters *CBfunction:* The far point address of the callback function module.
Return Value None
Example

```
Void CallBackFunction(void)
        {
                DWORD  dwCommEvent;
                dwCommEvent = wCi_GetCommEvent();
                // does something
        }
        void Init(void)
{
        ...
// Register Call Back
                FARPROC CBfunction = (FARPROC)(CallBackFunction);
                RegisterCallBack(CBfunction);
}
```

wCi_GetRegister

Call this API function to get the data byte from the hardware register.

Syntax unsigned char wCi_GetRegister (unsigned char *Register*)
Parameters *Register:* The register address
Return Value Returns the content of the register in byte.

wCi_GetRegister16

Call this API function to get the data WORD from the hardware register.

Syntax unsigned int wCi_GetRegister16 (unsigned char *Register*)
Parameters *Register:* The register address
Return Value Returns the content of the register in WORD

wCi_SetRegister

Call this API function to write the data byte to the hardware register.

Syntax void wCi_SetRegister (unsigned char *Register*, unsigned char *Value*)
Parameters *Register:* The register address.
Value: The value, which is written to the register

Return Value None

wCi_SetRegister16

Call this API function to write the data WORD to the hardware register.

Syntax	void wCi_SetRegister16 (unsigned char Register)
Parameters	*Register:* The register address.
Return Value	Returns the content of the register in WORD.

wCi_SetMaskRegister

Call this API function to write the data value to the hardware mask register.

Syntax	void wCi_SetMaskRegister (unsigned char Mask)
Parameters	*Mask:* The mask value.
Return Value	None

wCi_SetSourceID

Call this API function to write the data value to the hardware Source ID register.

Syntax	void wCi_SetSourceID (unsigned char *SourceID*)
Parameters	*SourceID:* The ID number for Source device.
Return Value	None

wCi_SetDestinationID

Call this API function to write the data value to the hardware Destination ID register.

Syntax	void wCi_SetDestinationID (unsigned char *Destination ID*)
Parameters	*Destination ID:* The ID number for destination device.
Return Value	None

wCi_SetVendorID

Call this API function to write the data value to the hardware Vendor ID register.

Syntax	void wCi_SetVendorID (unsigned char *VendorID*)
Parameters	*VendorID:* The ID number for Vendor device.
Return Value	None

wCi_SetNetworkID

Call this API function to write the data value to the hardware Network ID register.

Syntax	void wCi_SetNetworkID (unsigned char *NetworkID*)
Parameters	*NetworkID:* The ID number for Network device.
Return Value	None

wCi_SetIntermediateID

Call this API function to write the data value to the hardware Intermediate ID register.

Syntax	void wCi_SetIntermediateID (unsigned char *IntermediateID*)
Parameters	*IntermediateID:* The ID number for Intermediate device.
Return Value	None

wCi_GetSourceID

Call this API function to read the data value from the hardware Source ID register.

Syntax	unsigned char wCi_GetSourceID (void)
Parameters	None
Return Value	The contents of the source ID register.

wCi_GetDestinationID

Call this API function to read the data value from the hardware Destination ID register.

Syntax	unsigned char wCi_GetDestinationID (void)
Parameters	None
Return Value	The contents of the destination ID register.

wCi_GetVendorID

Call this API function to read the data value from the hardware Vendor ID register.

Syntax	unsigned char wCi_GetVendorID (void)
Parameters	None
Return Value	The contents of the Vendor ID register.

wCi_GetNetworkID

Call this API function to read the data value from the hardware Network ID register.

Syntax	unsigned char wCi_GetNetworkID (void)
Parameters	None
Return Value	The contents of the Network ID register.

wCi_GetIntermediateID

Call this API function to read the data value from the hardware Intermediate ID register.

Syntax	unsigned char wCi_GetIntermediateID (void)
Parameters	None
Return Value	The contents of the Intermediate ID register.

wCi_GetStatusRegister

Call this API function to read the data value from the hardware Status register.

Syntax	unsigned char wCi_GetStatusRegister (void)
Parameters	None
Return Value	The contents of the Status register.

wCi_GetMaskRegister

Call this API function to read the data value from the hardware Mask register.

Syntax	unsigned char wCi_GetMaskRegister (void)
Parameters	None
Return Value	The contents of the Mask register.

wCi_GetRxDestinationID

Call this API function to read the destination ID in the session mode.

Syntax	unsigned char wCi_GetDestinationID (void)
Parameters	None
Return Value	The Destination ID

wCi_InitiateSession

Call this API function to request the data session.

Syntax void wCi_InitiateSession (void)
Parameters None
Return Value None

wCi_AbortSession

Call this API function to abort the data session.

Syntax void wCi_AbortSession (void)
Parameters None
Return Value None

wCi_TransceiverReset

Call this API function to reset the data transmit/receive process.

Syntax void wCi_TransceiverReset (void)
Parameters None
Return Value None

wCi_SetBarCodePrinterID

Call this API function to set the ID for the printer.

Syntax void wCi_SetBarCodePrinterID (unsigned char *ID*, BOOL *flag*)
Parameters *ID:* The printer device ID.
Flag: Enable (TRUE) or disable (FALSE) the printer.
Return Value None

wCi_GetBarCodePrinterID

Call this API function to get the ID for the printer.

Syntax void wCi_GetBarCodePrinterID (unsigned char* *ID*, BOOL* *flag*)
Parameters *ID:* Points to the buffer that receives printer ID.
Flag: Points to the buffer that receives printer status.
Return Value None

wCi_StoreZeusParameters

Call this API function to store the current setup as the ZEUS parameters in EEPROM.

Syntax void wCi_StoreZeusParameters (void)
Parameters None
Return Value None

wCi_StoreOemParameters

Call this API function to store the current setup as the OEM parameters in EEPROM.

Syntax void wCi_StoreOemParameters (void)
Parameters None
Return Value None

wCi_StoreUserParameters

Call this API function to store the current setup as the USER parameters in EEPROM.

Syntax void wCi_StoreUserParameters (void)
Parameters None
Return Value None

wCi_StorePowerupParameters

Call this API function to store the current setup as the Power up parameters in EEPROM.

Syntax void wCi_StorePowerupParameters (void)
Parameters None
Return Value None

wCi_LoadZeusParameters

Call this API function to restore the ZEUS parameter setup.

Syntax void wCi_LoadZeusParameters (void)
Parameters None
Return Value None

wCi_LoadOemParameters

Call this API function to restore the OEM parameter setup.

Syntax void wCi_LoadOemParameters (void)
Parameters None
Return Value None

wCi_LoadUserParameters

Call this API function to restore the User parameter setup.

Syntax void wCi_LoadUserParameters (void)
Parameters None
Return Value None

wCi_LoadPowerupParameters

Call this API function to restore the Power up parameter setup.

Syntax void wCi_LoadPowerupParameters (void)
Parameters None
Return Value None

wCi_SetString

Call this API function to write the string data to string alias ID.

Syntax void wCi_SetString (unsigned char *Addr*, unsigned char* *Buffer)*
Parameters *Addr:* The String alias ID.
Buffer: Points to the data buffer.
Return Value

wCi_GetString

Call this API function to read the data from the string alias.

Syntax unsigned char* wCi_GetString (unsigned char *Addr*, unsigned char* *Buffer)*
Parameters *Addr:* The address of the String.
Buffer: Points to the buffer that receives the string alias characters.
Return Value Returns the pointer to the string data.

Remote Transceiver API

wCi_RemoteGetRegister

Call this API function to get the data from one or more registers.

Syntax	wCi_RemoteGetRegister (*r*)
Parameters	*Register*: The list of register address(es)
Return Value	Returns the content of the register(s).

<u>wCi_RemoteSetRegister</u>

Call this API function to write data to one or more hardware registers. The data are in register – value pairs.

Syntax	void wCi_RemoteSetRegister (unsigned char *Register*, unsigned char *Value*)
Parameters	*Register*: The register address. *Value*: The value, which is written to the register
Return Value	None

<u>wCi_RemoteGetConfig</u>

Call this API function to request the configuration of a remote transceiver.

Syntax	wCi_RemoteGetConfig ()
Parameters	
Return Value	Returns the.

<u>wCi_RemoteSetConfig</u>

Call this API function to set the configuration of a remote transceiver.

Syntax	wCi_RemoteSetConfig ()
Parameters	
Return Value	Returns the.

<u>wCi_RemoteGetStatus</u>

Call this API function to request the status of a remote transceiver.

Syntax	wCi_RemoteGetStatus ()
Parameters	
Return Value	Returns the.

<u>wCi_RemoteGetFullStatus</u>

Call this API function to request the full status of a remote transceiver.

Syntax wCi_RemoteGetFullStatus ()
Parameters
Return Value Returns the.

Communications API

Tx_Attempt_Limit	the maximum number of times to attempt a retransmission. 0xFF is infinite which is the default.
Request_RT_Comm	requests the current state of a remote transceiver's communications bytes in TX queue bytes in RX queue session holdoff enabled/disabled max no data hops max bad hops master/slave threshold bytes threshold time encryption on/off

RF API

Request_RT_RF	requests the current state of a remote transceiver's RF temperature oscillator temperature adjustment (TCXO) TX power sleep mode enabled/disabled break frequency 1 break frequency 2 break frequency 3

RS232 Signaling API

Request_RT_RS232	requests the current state of a remote transceiver's RS232 connection data rate flow control power on message enabled/disabled

Configuration API

Save_User_Config	saves the user configuration data to EEPROM
Restore_User_Config	restore the user configuration data from EEPROM
Read_User_Config	read the user configuration data from EEPROM

Save_OEM_Config	saves the OEM configuration data to EEPROM
Restore_OEM_Config	restore the OEM configuration data from EEPROM
Read_OEM_Config	read the OEM configuration data from EEPROM
Save_Factory_Config	saves the Zeus factory configuration data from EEPROM
Restore_Factory_Config	restore the Zeus factory configuration data from EEPROM and ROM
Read_Factory_Config	read the Zeus factory configuration data from EEPROM and ROM
Request_RT_Config	requests the current state of a remote transceiver's configuration power-on configuration (the user configuration) OEM configuration

Diagnostics API

Perform_Self_Test	put a specific node in self test
Verify_EEPROM	performs CRC checks on the EEPROM data
Clear_Error_Counts	clears the error counts in the adjacent node table (Bad hop count)
Set_Test_Data	allows the user to specify what data to send in a test message.
Send_Test_Message	sends a test message of n bytes to a specific node at rate y
Stop_Test_Message	stops sending test message.
Host_Loop_Back	loops back any data received from the host
Request_RT_Diagnostics	requests the current state of a remote transceiver's diagnostics EEPROM status bad hop count average RSSI current RSSI quality of service

Network Management API

Send_Vector_Table	distributes the vector table to a node
Request_Vector_Table	request the vector table from a specific node
Modify_Vector_Table	manually change a vector table
Update_Vector_Tables	updates vector tables form routing table changes and returns which have changed
Recover_Route_Table	request the vector tables from all directly connected nodes
Display_Route_Table	causes the routing table to be displayed
Display_Node_Status	returns node status table
Check_Node_Status	requests status from a specific node
Request_Periodic_Status	request that a node send its status every n minutes
Poll_Node_Status	poll each node for status every n minutes
Request_Adjacent_Nod	request the adjacent node table from a specific node

es	
Mark_Node_Inactive.	marks a node inactive in the routing table
Mark_Node_Active	marks a node active in the routing table
Alarm_Attributes	specifies what to do when an alarm occurs
Set_Alarm_Thresholds	set the number of error which must occur for an alarm to be generated
Set_Alarm_Email_Address	set e-mail address to notify in the event of an alarm
Set_Alarm_Page_Num	set number to page in the event of an alarm
Set_Alarm_Phone_Num	set number to phone in the event of an alarm
Send_Alarm_Email	send e-mail with alarm message n
Send_Alarm_Page	send page with alarm message n
Send_Alarm_Phone	make phone call with alarm message n
Locally_Display_Alarm	display alarm on operator's console with alarm attributes
Stop_Node_Tx	make a specific node go silent
Request_RT_Network	requests the current state of a remote transceiver's network view vector table adjacent node table

<u>Hop Table API</u>

Change_Hop_Table	change the nth entry in the hop table (value * 100 kHz)
Set_First_Channel	set which position in the hop table to use as the first channel
Set_Last_Channel	set which position in the hop table to use as the last channel
Set_CurrentChannel	set the current channel tuning
Get_CurrentChannel	get the current channel tuning
Set_Reference	set the IF and RF reference frequency divider
Select_Hop_Table	valid range is 1 - 100
Request_RT_Hop_Table	requests the current state of a remote transceiver's hop table hop table selected hop table reference frequency divider

It will be appreciated by persons of ordinary skill in the art that the present invention makes available an economical, compact frequency hopping spread spectrum wireless data telemetry transceiver network which includes .

Having described preferred embodiments of a new and improved circuit and method, it is believed that other modifications, variations and changes will be suggested to those skilled in the art in view of the teachings set forth herein. It is

therefore to be understood that all such variations, modifications and changes are believed to fall within the scope of the present invention as defined by the appended claims.

WHAT IS CLAIMED IS:

1) An on-air store and forward protocol method for dynamically establishing and maintaining a plurality of communication links between at least first, second and third spread spectrum frequency hopping transceivers designated as nodes and a host node, each including a memory pre-programmed with a plurality of pre-assigned frequencies, comprising the steps of:

a) in the first transceiver, selecting a first transmit frequency from the pre-assigned frequencies;

b) in the first transceiver, transmitting, on said first transmit frequency, a join network request;

c) in the second transceiver, receiving said join network request on said first transmit frequencies from said first transceiver;

d) in the second transceiver, transmitting a join network response in response to having received said first transceiver join network request;

e) in the third receiver, receiving said join network request on said first transmit frequencies from said first transceiver;

f) in the third receiver, transmitting a join network response in response to having received said first transceiver join network request;

g) in the first transceiver, receiving said second transceiver join network response and adding said second transceiver to a first transceiver adjacent node table in response thereto;

h) in the first transceiver, receiving said third transceiver join network response and adding said third transceiver to said first transceiver adjacent node table in response thereto.

2) The on-air store and forward protocol method of claim 1, further comprising: i) in the first transceiver, transmitting to contents of said first transceiver adjacent node table to the host.

3) The on-air store and forward protocol method of claim 3, further comprising:

j) In the host, generating a route table including the contents of said first transceiver adjacent node table.

4) The on-air store and forward protocol method of claim 1, further comprising:

j) In the host, transmitting said route table including the contents of said first transceiver adjacent node table to said first transceiver.

5) The on-air store and forward protocol method of claim 1, further comprising:

k) In the host, transmitting said route table including the contents of said first transceiver adjacent node table to said second transceiver.

6) The on-air store and forward protocol method of claim 1, further comprising:

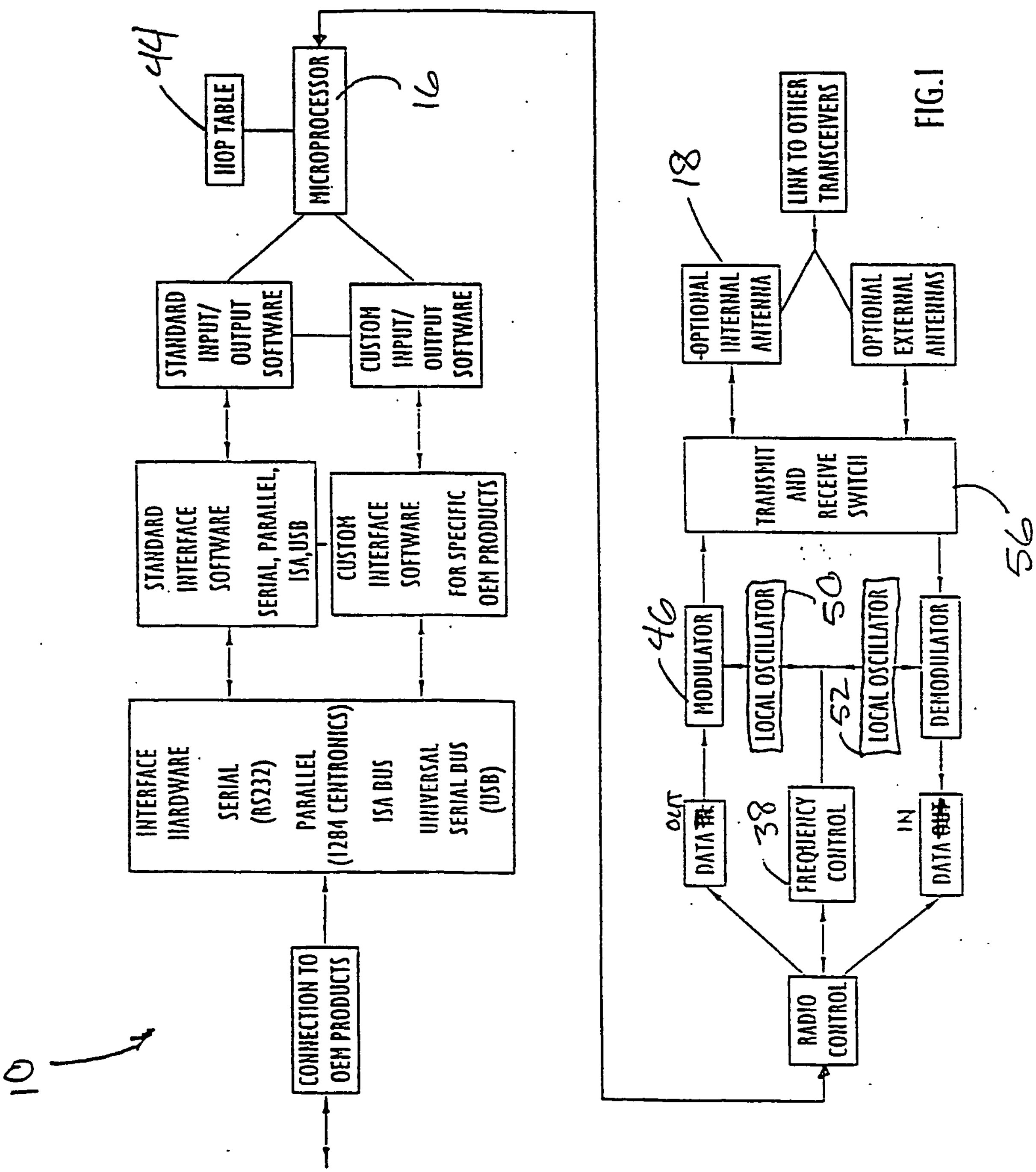
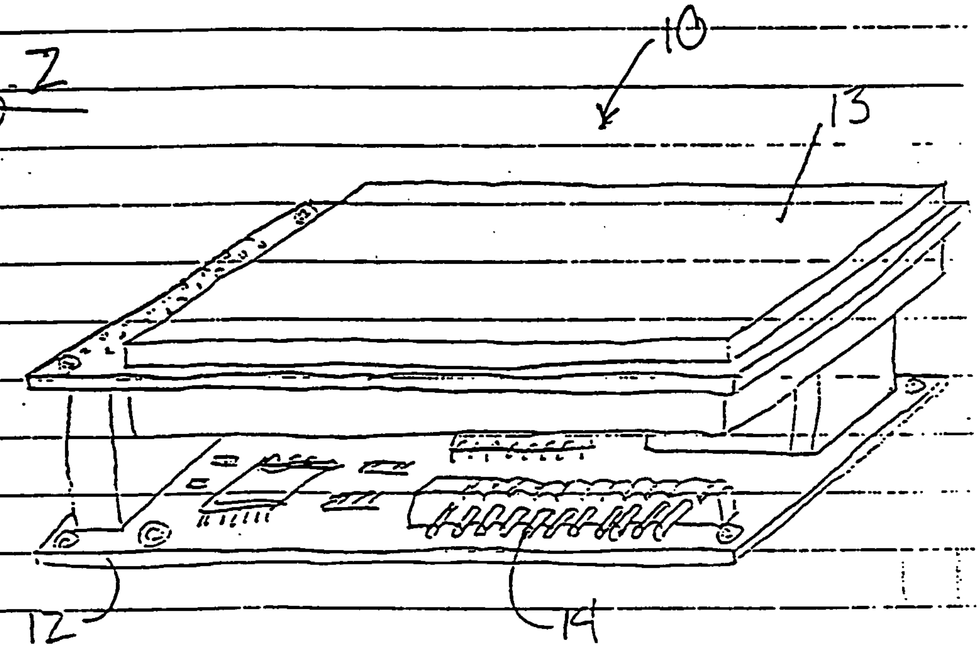

10
CONNECTION TO OEM PRODUCTS
INTERFACE HARDWARE
SERIAL (RS232)
PARALLEL (1284 CENTRONICS)
ISA BUS
UNIVERSAL SERIAL BUS (USB)
STANDARD INTERFACE SOFTWARE
SERIAL, PARALLEL, ISA,USB
CUSTOM INTERFACE SOFTWARE
FOR SPECIFIC OEM PRODUCTS
STANDARD INPUT/ OUTPUT SOFTWARE
CUSTOM INPUT/ OUTPUT SOFTWARE
MICROPROCESSOR
16
HOP TABLE
44
RADIO CONTROL
DATA OUT
DATA IN
FREQUENCY CONTROL
38
MODULATOR
46
LOCAL OSCILLATOR
50
LOCAL OSCILLATOR
52
DEMODULATOR
TRANSMIT AND RECEIVE SWITCH
56
OPTIONAL INTERNAL ANTENNA
18
OPTIONAL EXTERNAL ANTENNAS
LINK TO OTHER TRANSCEIVERS

FIG.1

Fig. 2



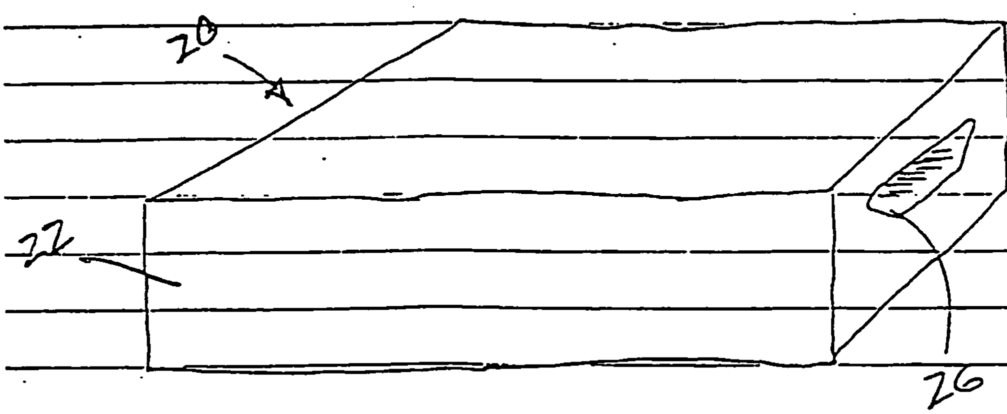


Fig. 3

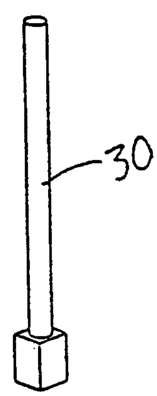


FIG.4B

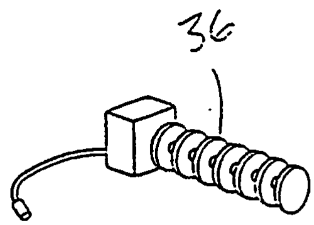


FIG.4A

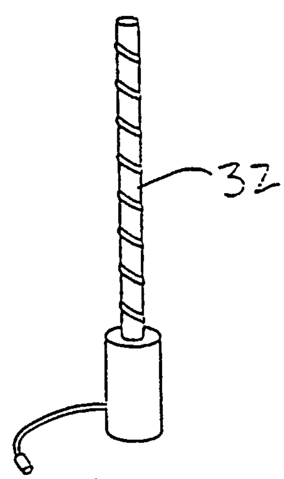


FIG.4C

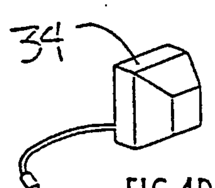


FIG.4D

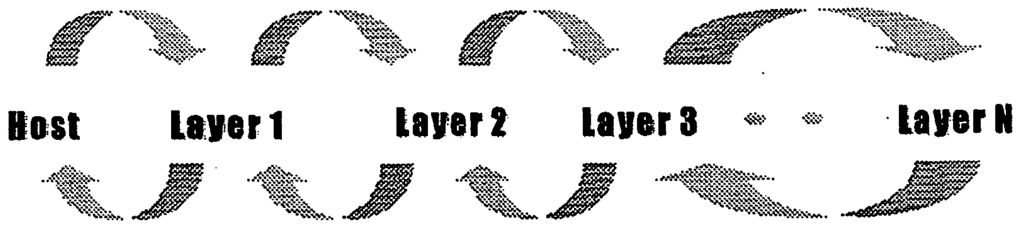


Figure 5 - Message Propagation

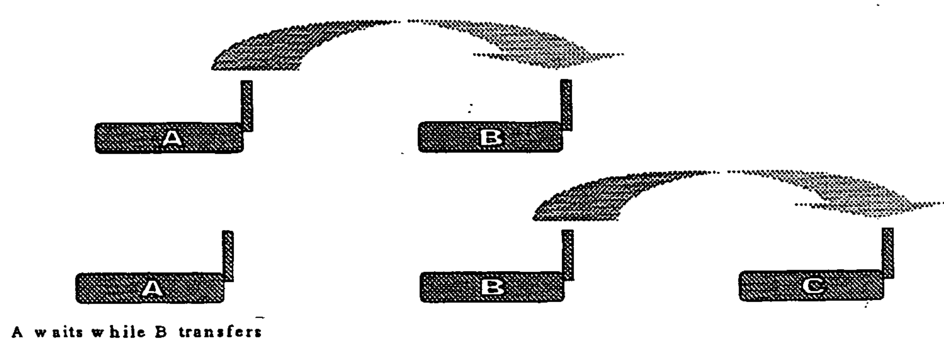


Figure 6

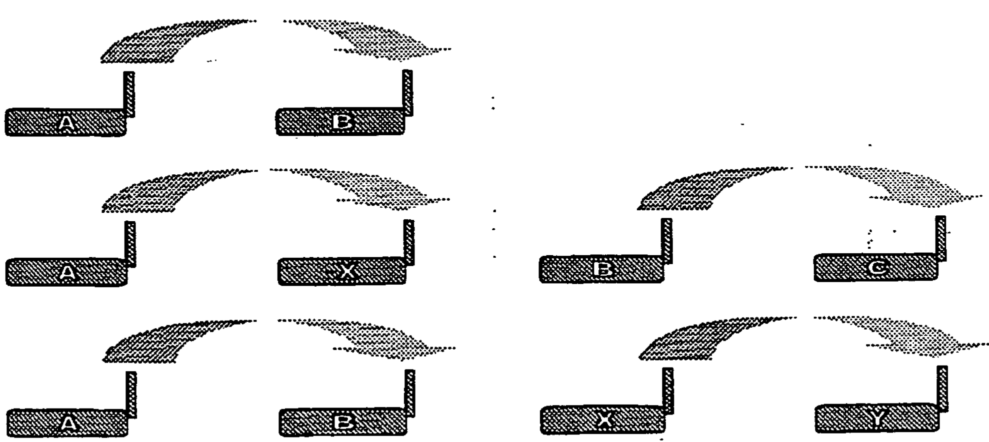


Figure 7

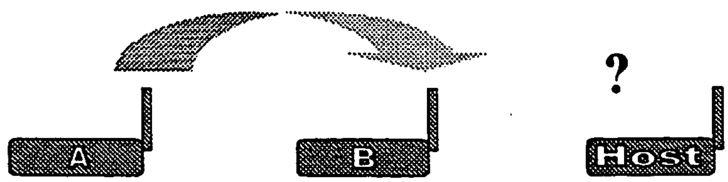


Figure 8

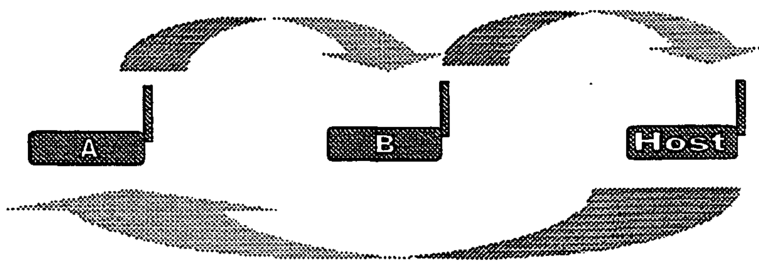


Figure 9

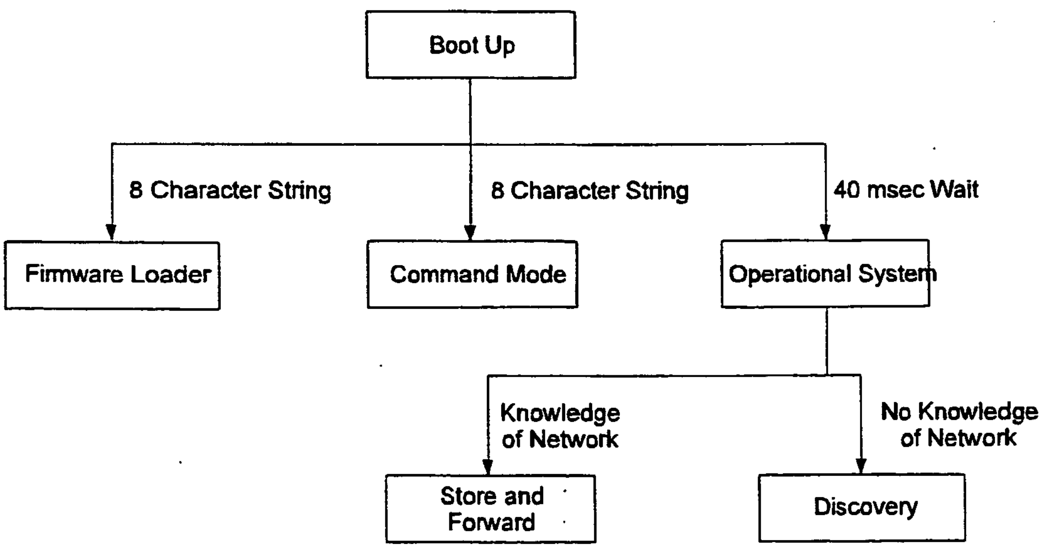


Figure 10

Join Net Request	announces a node which wishes to join the network
Join Net Response	the response by nodes which heard the request
Host Join Net	informs nodes which hear the message that they have a connection to the host
Host Join Net Response	the response by nodes which heard the request
Node Registration Request	the message used by a node to register its presence with the host
Node Registration Response	the acknowledgment by the host that the node is now registered
Network in Service	places the network in service for data transmission
Network out of Service	takes the network out of data transmission service
Vector Table Request	request an update of the vector table
Vector Table Update	an update of a nodes vector table from the host

Figure 11

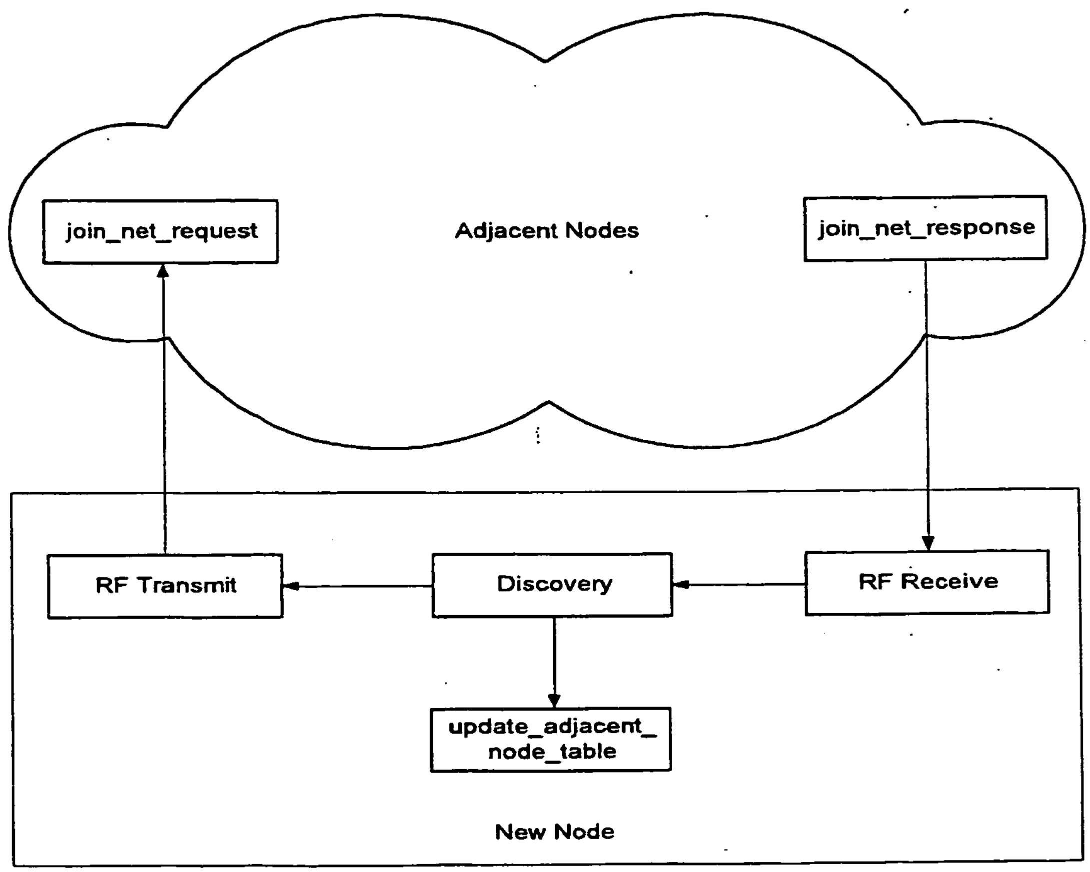


Figure 12

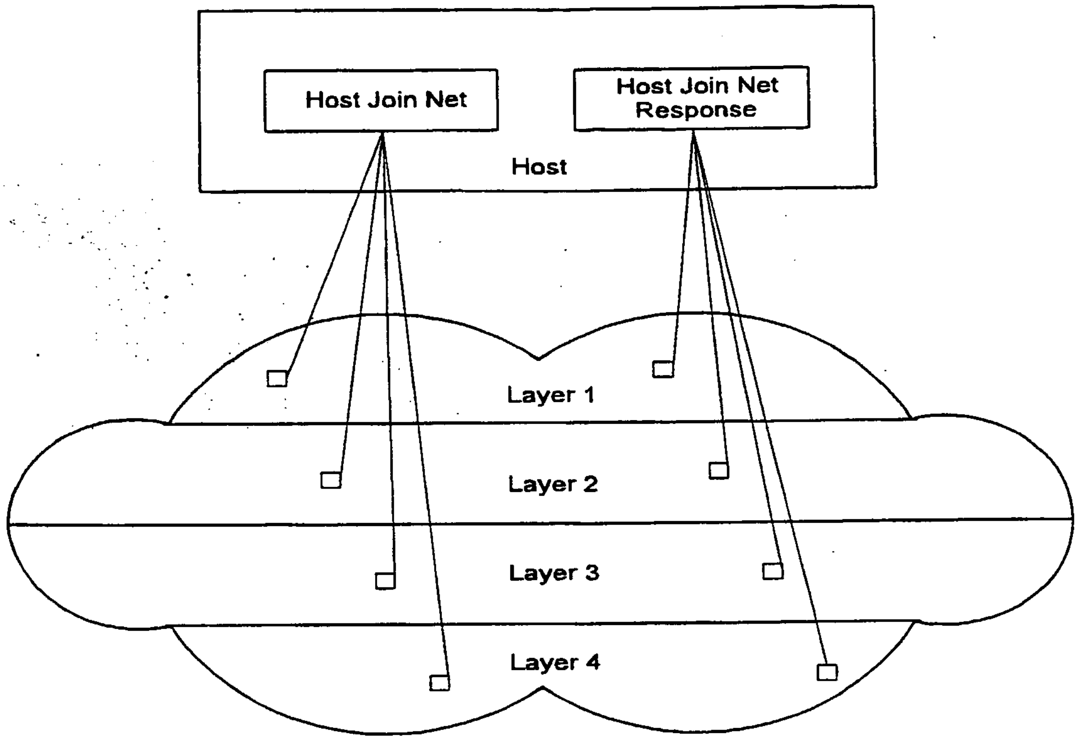


Figure 13

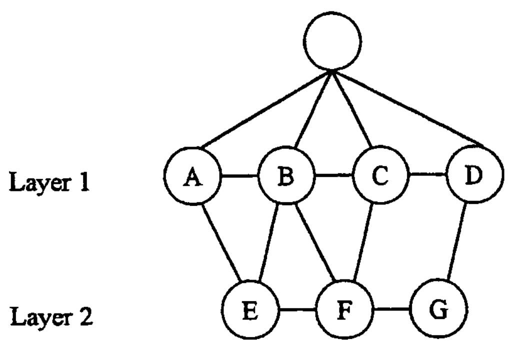


Figure 14

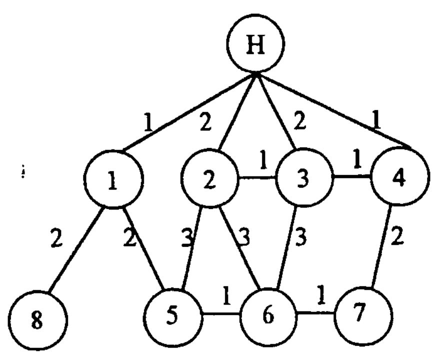


Figure 15

Node	PRIMARY	HOPS	ALTERNATE	HOPS
0	0	1	5	3
1	254	0	254	0
2	5	2	5	2
3	5	3	5	3
4	5	4	5	4
5	5	1	5	1
6	5	2	5	2
7	5	3	5	3
8	8	1	8	1

Figure 16

Node	PRIMARY	HOPS	ALTERNATE	HOPS
0	0	1	3	2
1	5	2	5	2
2	254	0	254	0
3	3	1	6	2
4	3	2	6	3
5	5	1	6	2
6	6	1	3	2
7	6	2	3	2
8	5	3	6	4

Figure 17

Node	PRIMARY	HOPS	ALTERNATE	HOPS
0	0	1	4	2
1	2	3	6	3
2	2	1	6	2
3	254	0	254	0
4	4	1	6	3
5	2	2	6	2
6	6	1	2	2
7	4	2	6	2
8	2	4	6	4

Figure 18

Node	PRIMARY	HOPS	ALTERNATE	HOPS
0	0	1	3	2
1	7	4	3	4
2	3	2	7	3
3	3	1	7	3
4	254	0	254	0
5	7	3	3	3
6	7	2	3	2
7	7	1	3	3
8	7	5	3	5

Figure 19

Node	PRIMARY	HOPS	ALTERNATE	HOPS
0	1	2	2	2
1	1	1	1	1
2	2	1	6	2
3	6	2	2	2
4	6	3	2	3
5	254	0	254	0
6	6	1	2	2
7	6	2	2	3
8	1	2	1	2

Figure 20

Node	PRIMARY	HOPS	ALTERNATE	HOPS
0	2	2	3	2
1	5	2	2	3
2	2	1	5	2
3	3	1	2	2
4	7	2	3	2
5	5	1	2	2
6	254	0	254	0
7	7	1	3	3
8	5	3	2	4

Figure 21

Node	PRIMARY	HOPS	ALTERNATE	HOPS
0	4	2	6	3
1	6	3	4	5
2	6	2	4	3
3	4	2	6	2
4	4	1	6	3
5	6	2	4	4
6	6	1	4	3
7	254	0	254	0
8	6	4	4	6

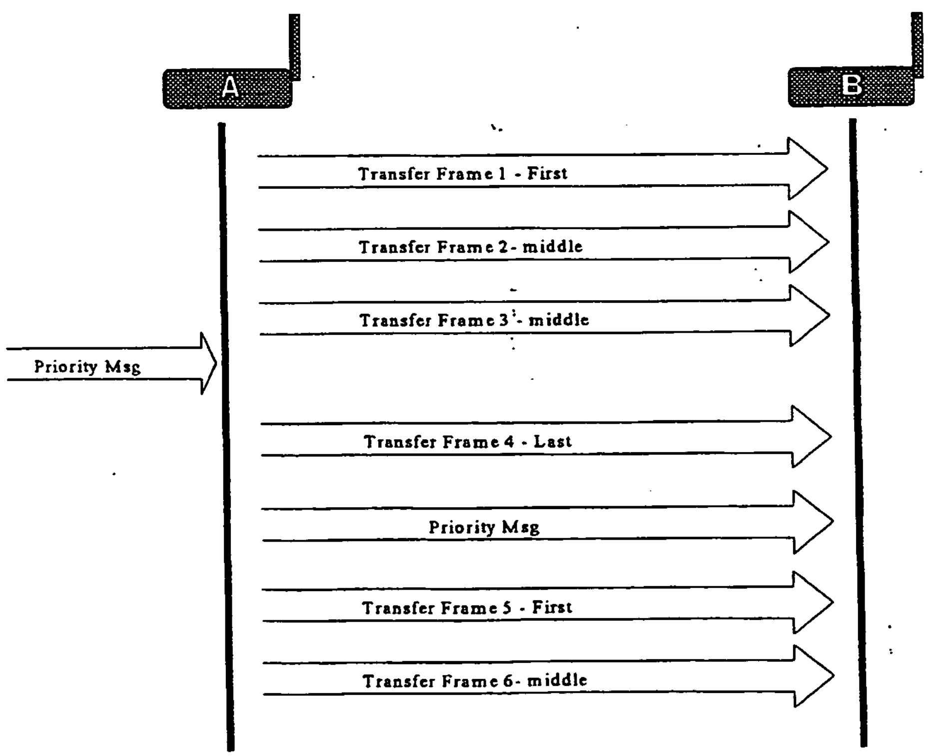
Figure 22

Node	PRIMARY	HOPS	ALTERNATE	HOPS
0	1	2	0	0
1	255	0	255	0
2	255	0	255	0
3	255	0	255	0
4	255	0	255	0
5	255	0	255	0
6	255	0	255	0
7	255	0	255	0
8	254	0	254	0

Figure 23

Message type	tells if this is a data packet, administrative packet, etc.	1 nibble
Vendor ID	the ID of the OEM reselling the ZLRT9600	2 bytes
Network ID	the unique network identifier	2 bytes
Destination ID	the ID of the final destination (set by the originator)	1 byte
intermediate ID	the node to send the data to next	1 byte
source ID	the ID of the originating node	1 byte
sequence number	modulo 128	1 byte
data position	01 – first data frame in stream 00 – middle of data stream 10 – last data frame 11 – both first and last frame, i.e., a single frame	2 bits
priority	true if this is a high priority packet	1 bit
ack type	0 – negative acknowledgement (NACK) included 1 – acknowledgement (ACK) included	1 bits
flow control	0 – clear to send 1 – stop sending	1 bit
TX power	the transmit power being used for this message	2 bits
RSSI	the signal strength of the last transmission from the intermediate node or a power up, stay the same, or power down indication	1 nibble
hop counter	the number of hops to the final destination	1 byte
data byte count	the number of data bytes in this frame	1 byte
CRC 16	the CRC of the header	2 bytes

Figure 24



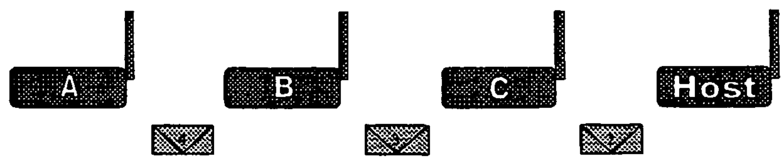


Figure 26

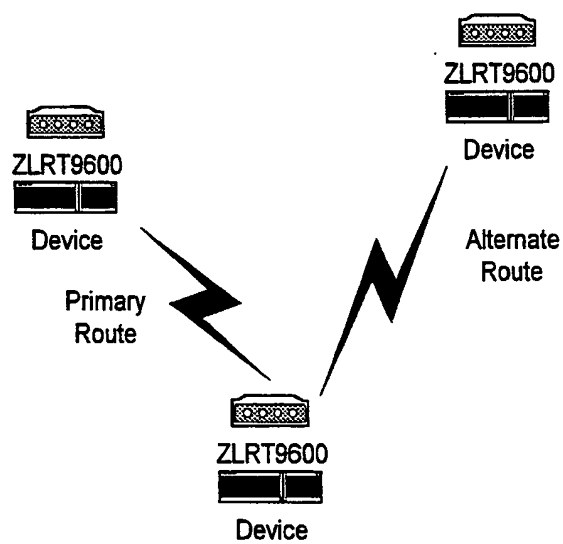


Figure 27

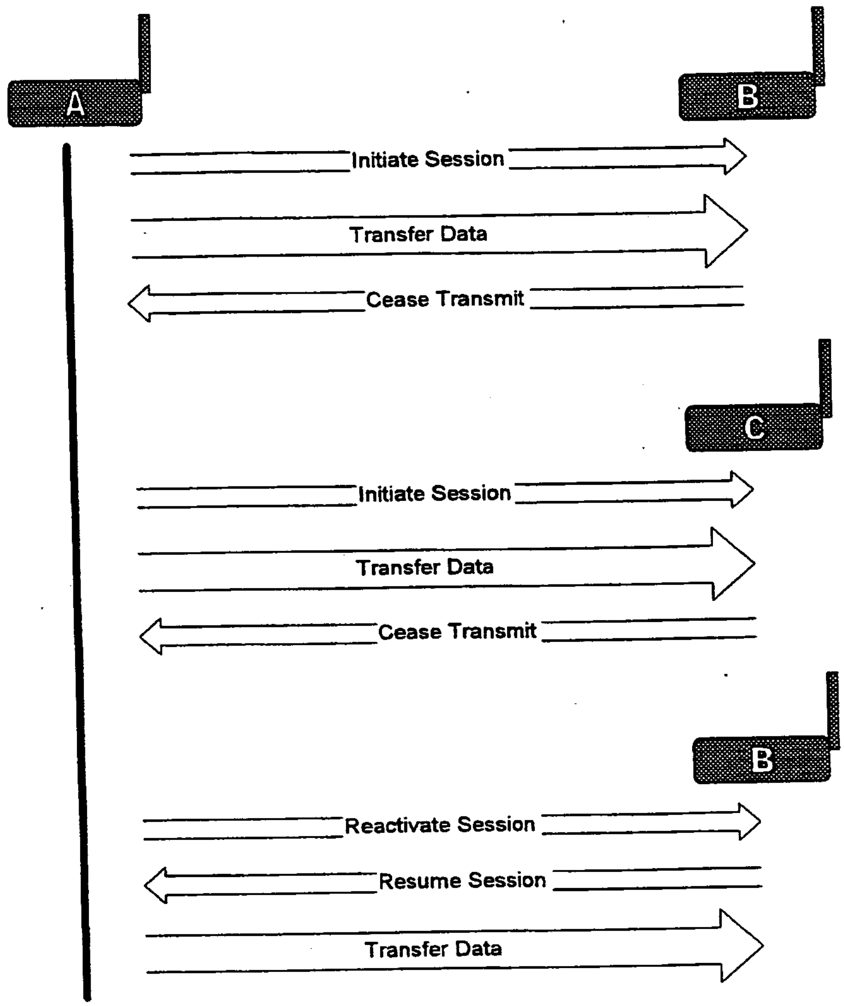


Figure 28

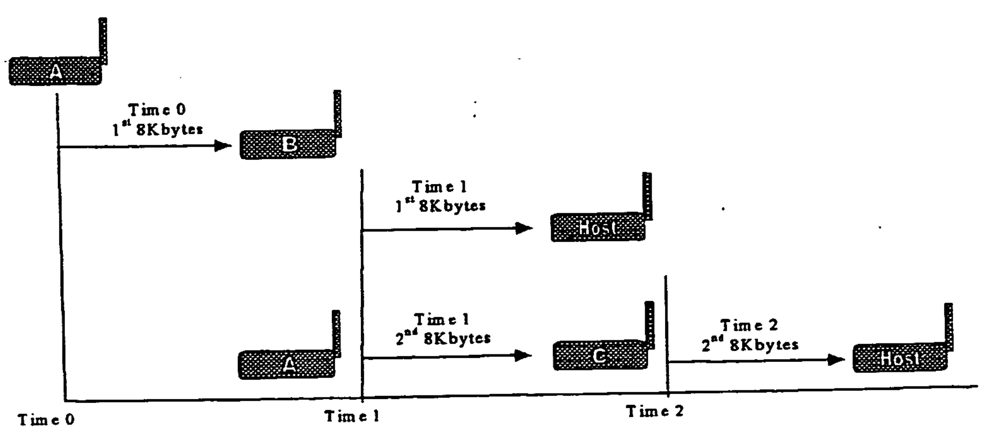


Figure 29

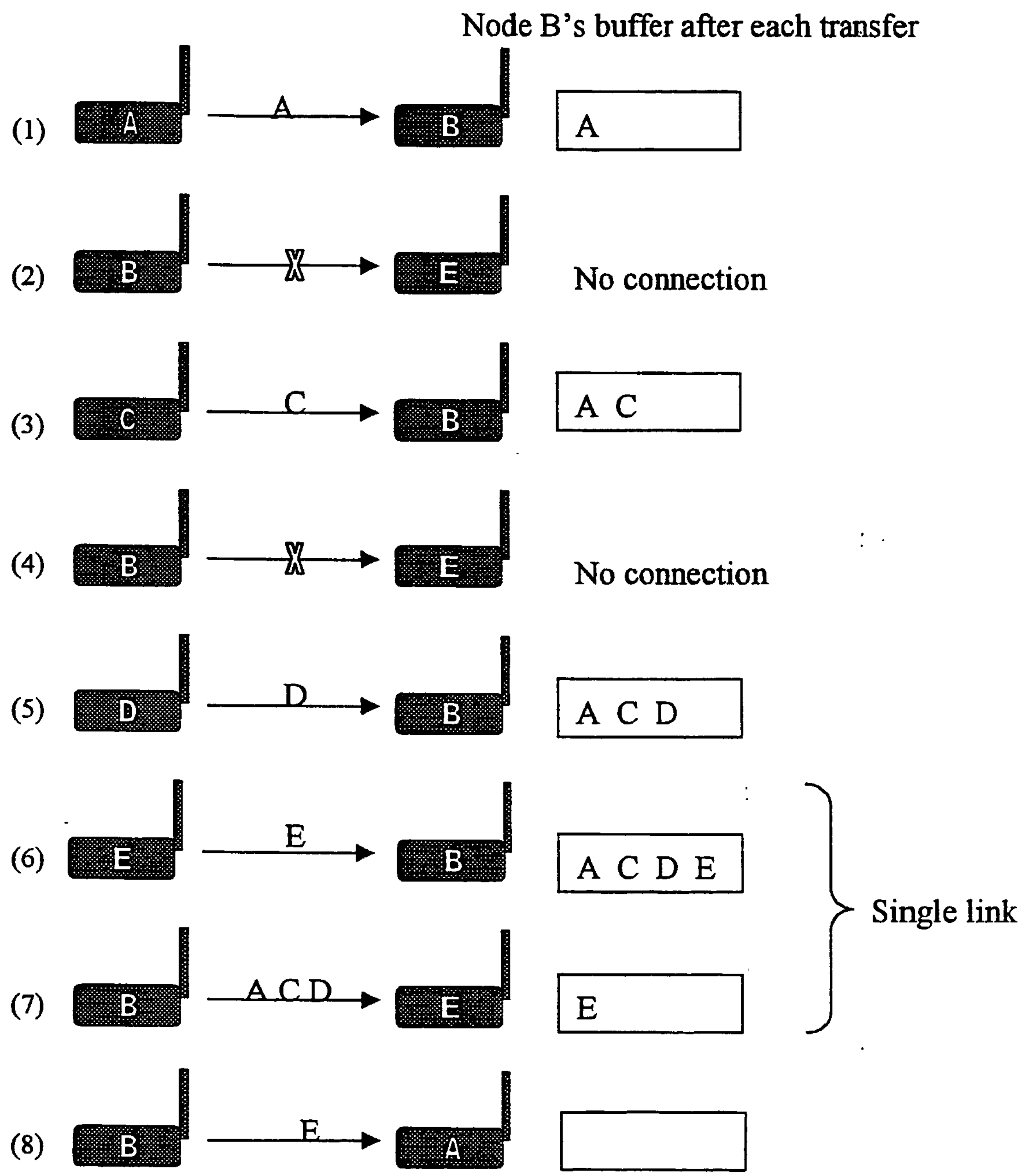

Node B's buffer after each transfer
(1)
A
A
B
A
(2)
B
X
E
No connection
(3)
C
C
B
A C
(4)
B
X
E
No connection
(5)
D
D
B
A C D
(6)
E
E
B
A C D E
(7)
B
A C D
E
E
Single link
(8)
B
E
A

Figure 30

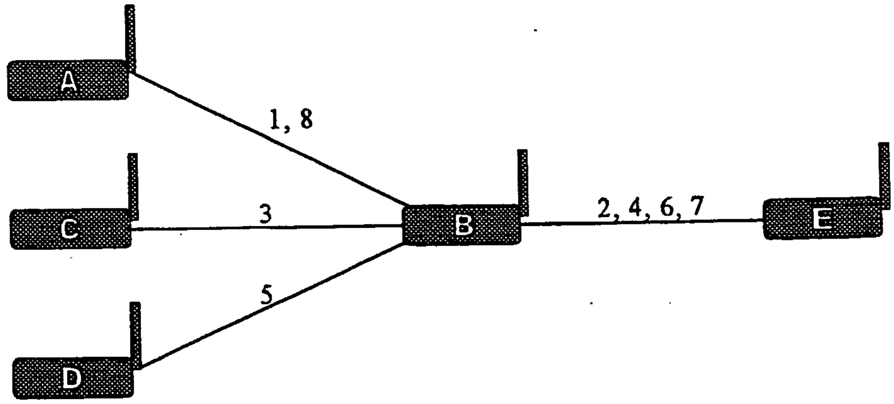


Figure 31

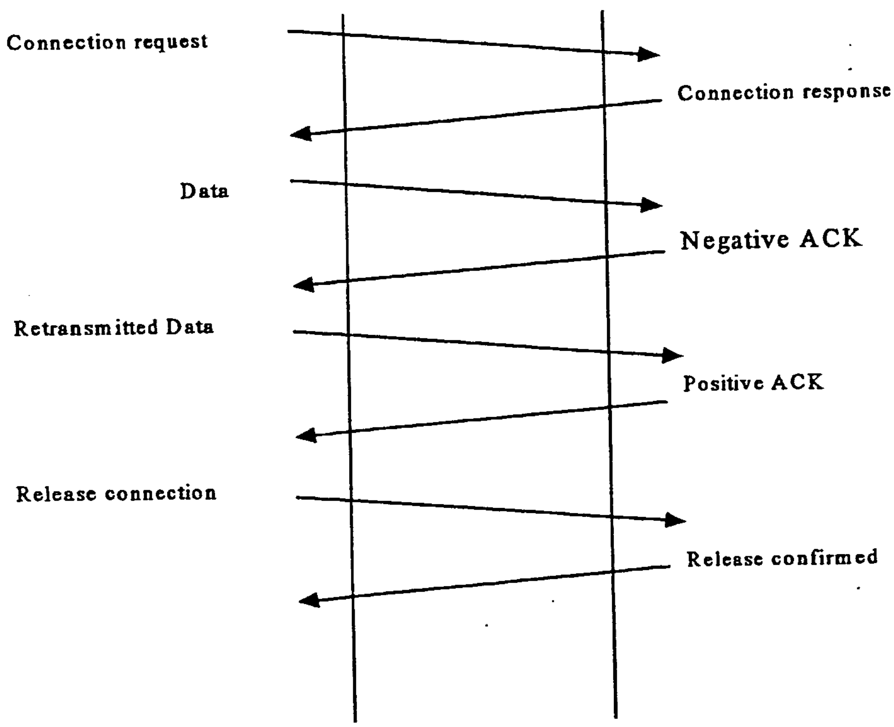


Figure 32

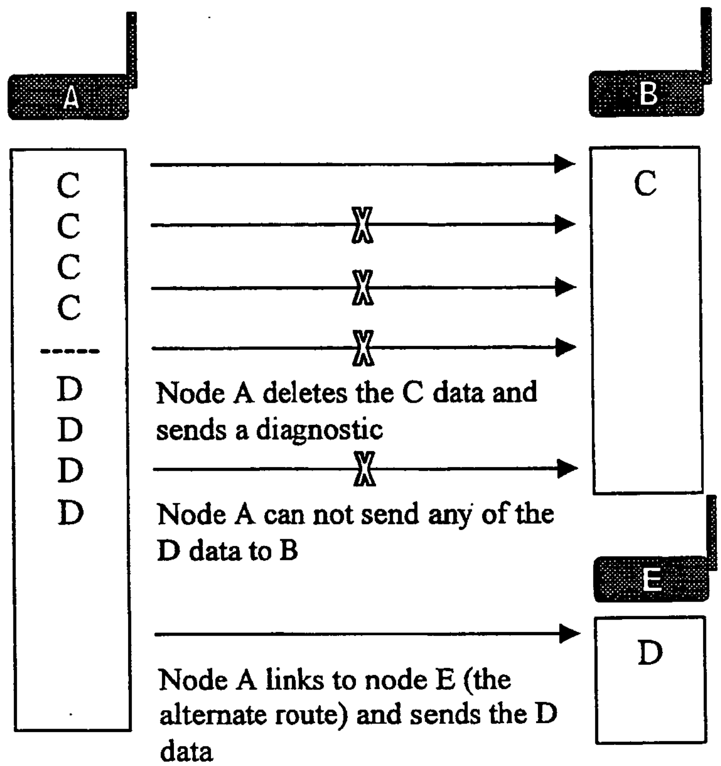


Figure 33

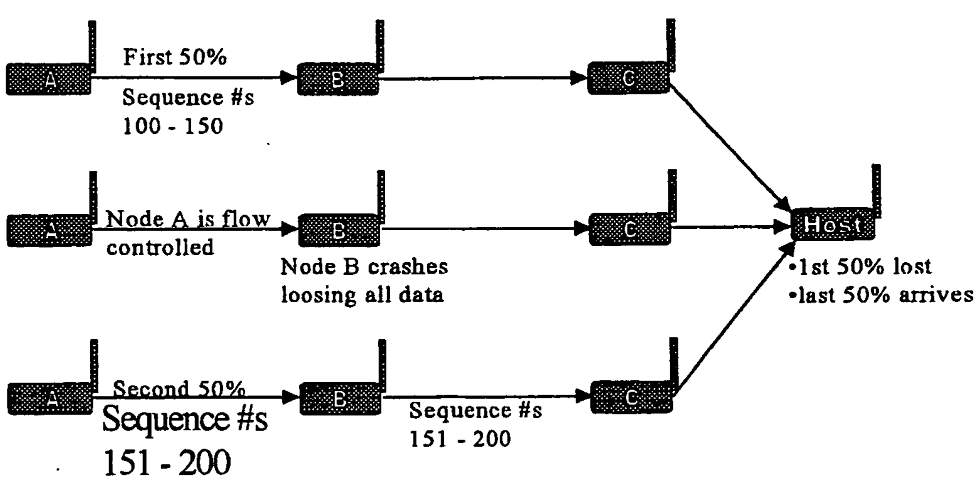


Figure 34

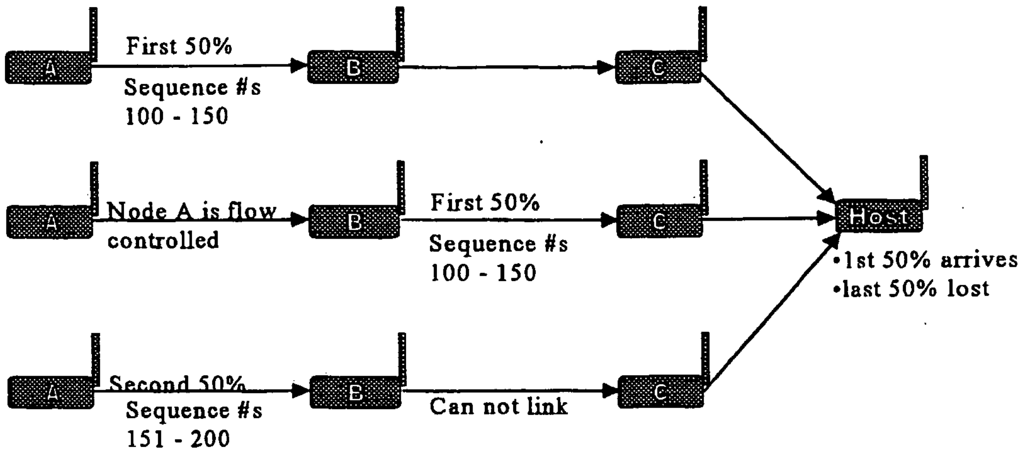


Figure 35

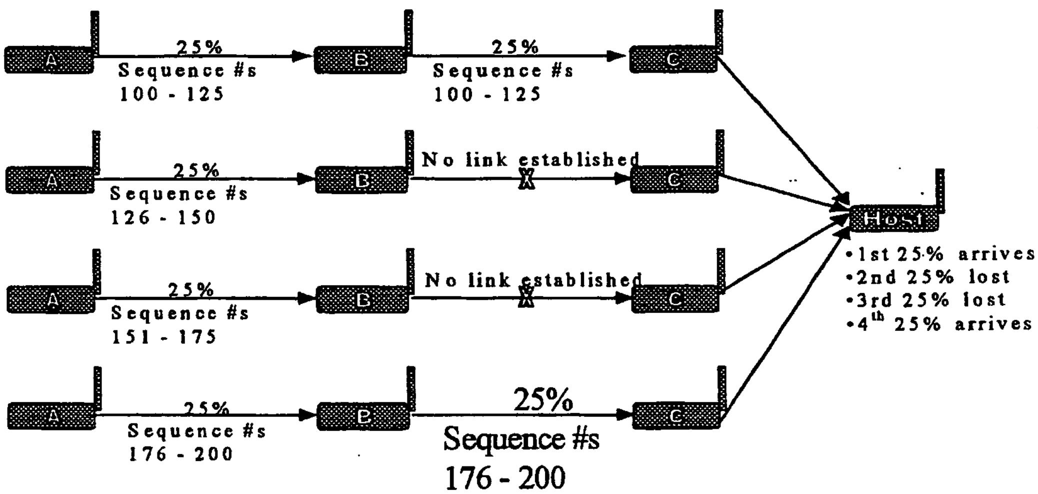


Figure 36

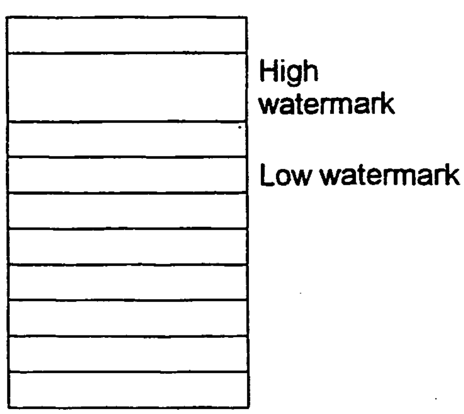


Figure 37

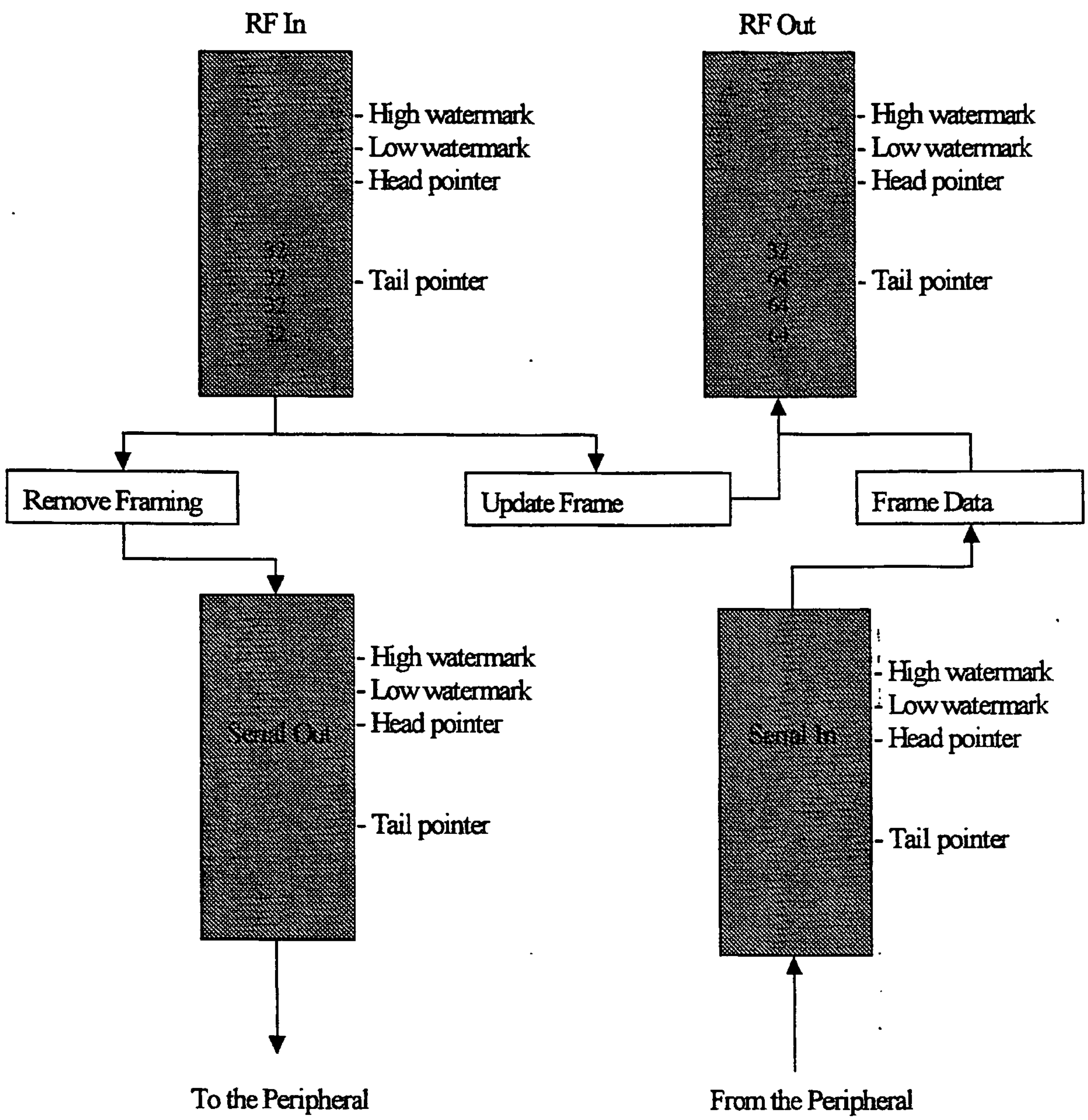

RF In
RF Out
- High watermark
- Low watermark
- Head pointer
- Tail pointer
Remove Framing
Update Frame
Frame Data
Serial Out
Serial In
To the Peripheral
From the Peripheral

Figure 38

Frame Header
Admin. Message Type
Admin. Message Number
Admin Message Contents
Admin. Message Number
Admin Message Contents
Admin. Message Number
Admin Message Contents
CRC16

Frame Header
Admin. Message Type
Admin. Message Number
Admin Message Size
Admin Message Contents
Admin. Message Number
Admin Message Size
Admin Message Contents
Admin. Message Number
Admin Message Size
Admin Message Contents
CRC16

Figure 39

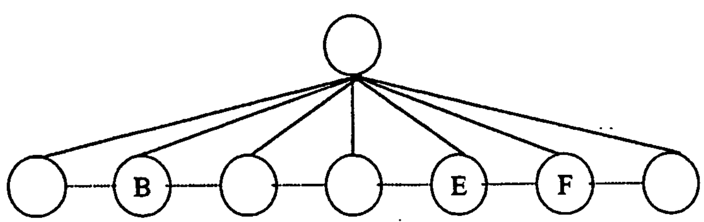


Figure 40

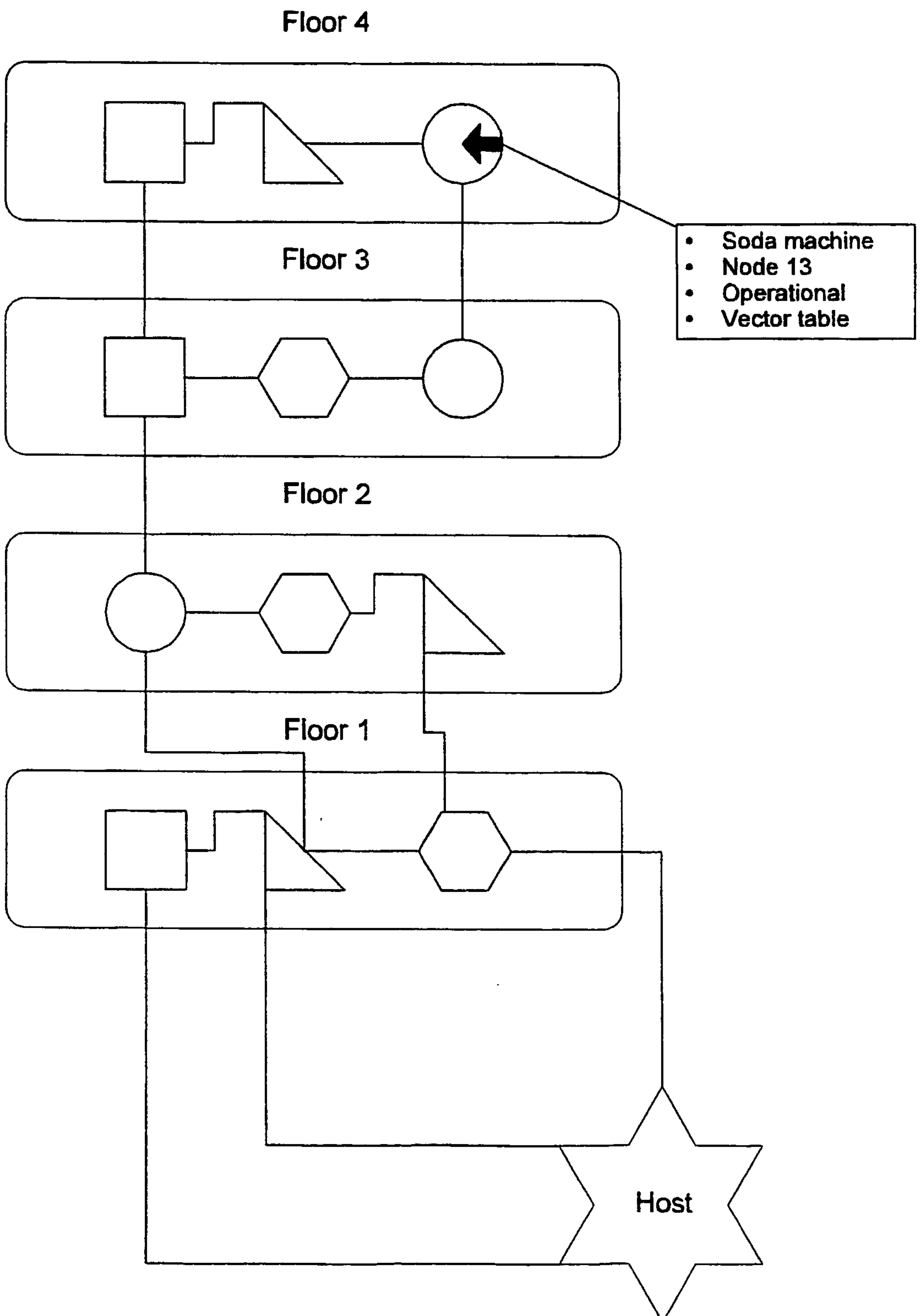

Floor 4
Floor 3
Floor 2
Floor 1
Soda machine
Node 13
Operational
Vector table
Host

Figure 41

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US00/30472

A. CLASSIFICATION OF SUBJECT MATTER
IPC(7) : H04L 27/30
US CL : 375/130
According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
U.S. : 375/130; 370/225, 227, 238, 396, 408; 714/57

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
Please See Continuation Sheet

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A,P	US 6,122,759 A (AYANOGLU et al.) 19 September 2000, ALL	1-6
A	US 5,805,593 A (BUSCHE) 08 September 1998, ALL	1-6

☐ Further documents are listed in the continuation of Box C. ☐ See patent family annex.

* Special categories of cited documents:

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier application or patent published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search 15 February 2001 (15.02.2001)	Date of mailing of the international search report 11 APR 2001
Name and mailing address of the ISA/US Commissioner of Patents and Trademarks Box PCT Washington, D.C. 20231 Facsimile No. (703)305-3230	Authorized officer Stephen Chin Eugenia Logan Telephone No. 703 305-3900

Form PCT/ISA/210 (second sheet) (July 1998)

Continuation of B. FIELDS SEARCHED Item 3: EAST
search terms: routing table, node, acknowledge, network